

***Indigo Systems
LabVIEW 7 Toolkit User's Guide
For Indigo Systems Cameras
Version 1.10
321-0026-10***

September, 2003

Indigo Systems Corp.

**Indigo Systems LabVIEW 7 Toolkit
Version 1.10**

If You Need Help – Technical Support

If you need help at any time with the installation or use of the *Indigo LabVIEW Toolkit*, please call Client Services at Indigo Systems:

(805) 964-9797

or visit our website at www.indigosystems.com and click on the Client Services button.

Disclaimer

In no event shall Indigo Systems or its representatives be liable for any consequential damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other loss), arising out of the use of or inability to use the *Indigo LabVIEW Toolkit*.

Copyright Trademarks and Version

Copyright © 2002 Indigo Systems, Corp. All rights reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written consent of, Indigo Systems Corp.

05/2002 Initial Release. This manual was updated to *Indigo LabVIEW Toolkit* version 1.0.

09/2003 Update to include operation with current NI software. See Section 1.1 for details.

Indigo Systems® is a registered trademark of Indigo Systems Corporation.

CONTENTS

1	INTRODUCTION	7
1.1.	HARDWARE AND SOFTWARE REQUIREMENTS	7
1.2.	INSTALLING THE INDIGO LABVIEW TOOLKIT.....	7
1.3.	OVERVIEW OF LIBRARIES IN THE INDIGO LABVIEW TOOLKIT.....	8
2	GETTING STARTED.....	9
2.1	INDIGO LABVIEW TOOLKIT OVERVIEW	9
2.2	EXAMPLES.....	9
2.2.1	 Ex. 1 Example 1.VI.....	10
2.2.2	 Ex. 2 Example 2.VI.....	12
2.2.3	 Ex. 3 Example 3.VI.....	14
2.2.4	 Ex. 3 Example 3.VI Modified.....	15
2.2.5	 Ex. 4 Example 4.VI.....	16
2.2.6	 Ex. 5 Example 5.VI.....	18
2.2.7	 Camera GUI.VI.....	21
3	TOOLKIT VI REFERENCE	28
3.1	INDIGO LABVIEW TOOLKIT MAIN.LLB.....	28
3.1.1	 Close Toolkit Library.VI	28
3.1.2	 Continuous Image Acquire.VI.....	29
3.1.3	 Histogram.VI.....	31
3.1.4	 Image Acquisition Setup.VI.....	32
3.1.5	 Image Compression.VI.....	34
3.1.6	 Image Compression Init.VI	36
3.1.7	 Image Display Setup.VI.....	37
3.1.8	 Invert_Revert.VI.....	38
3.1.9	 Load Palette	39

Indigo Systems LabVIEW 7 Toolkit Version 1.10

3.1.10		Polarity.VI.....	40
3.1.11		Remove Edges.VI.....	41
3.1.12		Save to Disk.....	43
3.1.13		Single Image.VI.....	44
3.1.14		Set Color Palette.VI	46
3.2		THERMOTEKNIX PALETTE UTILITIES.LLB	47
3.2.1		Load Thermoteknix Palette GUI.VI	48
3.2.2		Load Thermoteknix Palette File.VI.....	49
3.2.3		Find Thermoteknix Palette Names.VI	50
3.2.4		Load Thermoteknix Color Palette by Name.VI.....	51
3.2.5		Construct IMAQ Palette.VI.....	52
3.2.6		Make Color Table.VI.....	53
3.3		NON-UNIFORMITY CORRECTION AND BAD PIXEL REPLACEMENT	54
3.3.1		Check BPM.VI.....	58
3.3.2		Check Gains.VI	59
3.3.3		Check Offsets.VI.....	60
3.3.4		Close NUC_BPR Library.VI.....	61
3.3.5		Initialize NUC and BPR.VI	62
3.3.6		NUC and BPR.VI	63
3.3.7		Read BPR from Settings File.VI.....	64
3.3.8		Read Gains from Settings File.VI.....	66
3.3.9		Read Offsets from Settings File.VI.....	67
3.3.10		Read Header Array Elements from File.VI.....	68
3.4		LOW-LEVEL TOOLKIT UTILITIES	69

Indigo Systems LabVIEW 7 Toolkit Version 1.10

3.4.1		Test Image.VI.....	69
3.4.2		Invert Palette.VI.....	70
3.4.3		Remove Integers from String.VI.....	71
3.4.4		Indigo Palettes Global.VI.....	72
3.5		OTHER LIBRARIES AND VIS IN THE TOOLKIT	73
4		ERROR CODES.....	73
5		CONCLUDING REMARKS.....	74
6		INTRODUCTION	76
6.1		HARDWARE AND SOFTWARE REQUIREMENTS	76
6.2		INSTALLING THE INDIGO ALPHA NIR LABVIEW TOOLKIT	76
6.3		INDIGO ALPHA NIR LABVIEW TOOLKIT OVERVIEW	78
7		GETTING STARTED.....	79
7.1		INDIGO ALPHA NIR LABVIEW TOOLKIT OVERVIEW	79
7.2		VERIFYING CAMERA OPERATION	79
7.3		TOP LEVEL VIS.....	79
7.3.1		Alpha NIR GUI.VI.....	80
7.3.2		Alpha NIR Control.VI.....	81
8		LOWER LEVEL AND OTHER VIS.....	83
8.1		LOWER LEVEL TOOLKIT VIS.	83
8.1.1		Bytes2Short.VI.....	84
8.1.2		Camera Write_Read.VI.....	84
8.1.3		Display FPA Register.VI.....	84
8.1.4		Image Acquisition Setup for Alpha.VI.....	85
8.1.5		Image Display Setup for Alpha.VI	85
8.1.6		Indigo.AlphaGlobal.VI.....	86
8.1.7		Interface Dialog.VI	86
8.1.8		Read camera settings.VI.....	86
8.1.9		Read Camera ID.VI.....	87

Indigo Systems LabVIEW 7 Toolkit Version 1.10

8.1.10		Read Camera IT.VI.....	88
8.1.11		Read Case Temperature.VI.....	88
8.1.12		Read Extended Integration Timer.VI.....	88
8.1.13		Read FPA Mode.VI.....	89
8.1.14		Read FPA Temp.VI.....	89
8.1.15		Read Int Timer.VI.....	89
8.1.16		Read long int mode.VI.....	90
8.1.17		Read short int timer.VI.....	90
8.1.18		Read VdetCom.VI.....	90
8.1.19		Receive Camera.VI.....	91
8.1.20		Select Gain Mode.VI, ...	91
8.1.21		Set Gain Mode.VI.....	91
8.1.22		Set IT.VI.....	92
8.1.23		Set IT with Error Check.VI.....	92
8.1.24		Set Orientation.VI.....	93
8.1.25		Short2bytes.VI.....	93
8.1.26		Status.VI.....	93
8.2	OTHER VIS IN THE TOOLKIT		94
9	ERROR CODES.....		94
10	CONCLUDING REMARKS.....		95

Introduction

Indigo LabVIEW Toolkit is a set of image acquisition and analysis subroutines and dlls for the LabVIEW/IMAQ programming environment. The Toolkit provides the user with acquisition, storage, retrieval, display, and processing of still images and video using Indigo Systems Cameras. It is assumed that the user is at least somewhat proficient with the LabVIEW/IMAQ programming environment. Additional information concerning LabVIEW and IMAQ training are available at the National Instruments web page at www.ni.com. Custom software development for selected Indigo Systems imaging products is available from WinSoft Corporation at www.winsoft.com. The block diagrams for many of the VIs in this toolkit are password protected.

1.1. Hardware and Software requirements

To install **Indigo LabVIEW Toolkit** you will need the following:

- 1.1.1 IBM compatible, Pentium based computer. Pentium III 800 MHz or better recommended.
- 1.1.2 XGA (1024 x 768) or better screen resolution.
- 1.1.3 Windows 2000 or Windows XP operating system.
- 1.1.4 100 MB of available hard drive space. 200 MB or more recommended.
- 1.1.5 256 MB RAM memory. 512 MB recommended.
- 1.1.6 National Instruments LabVIEW 7.0 Express or higher full development version.
- 1.1.7 National Instruments IMAQ version 2.6 or higher and IMAQ Vision Development Module version 7.0 or higher. In order to run the IEEE 1394 examples, the NI IMAQ 1394 Driver Version 1.5 or higher is required.
- 1.1.8 *National Instruments PCI-1422 Frame Grabber Card.
- 1.1.9 *A supported Indigo Systems camera.
- 1.1.10 *National Instruments cable for the Indigo Systems camera.

not required for **Indigo LabVIEW Toolkit installation, but needed for full operation of the software.*

1.2. Installing the Indigo LabVIEW Toolkit

Indigo LabVIEW Toolkit is provided on a CD.

To install, simply copy the Indigo LabVIEW Toolkit folder to a convenient location on your hard drive. Right-click on each of the files and deselect "read only." Be sure National Instruments LabVIEW, IMAQ, and IMAQ Vision have been installed on the target machine. The toolkit should be installed on the same logical drive as IMAQ and IMAQ Vision.

1.3. Overview of Libraries in the Indigo LabVIEW Toolkit

Indigo LabVIEW Toolkit is a set of image acquisition and analysis subroutines and dlls for the LabVIEW/IMAQ programming environment that provides the user with acquisition, storage, retrieval, display, and processing of still images and video using supported Indigo Systems Cameras. It is assumed that the user is at least somewhat proficient with the LabVIEW/IMAQ programming environment. Diagrams for many of the VIs in the toolkit are password protected.

Chapter 2 contains a tutorial featuring a tour of several example applications developed using Indigo LabVIEW Toolkit VIs.

Chapter 3 contains a description of each VI in the Toolkit.

Chapter 4 lists error codes specific to the Indigo LabVIEW Toolkit.

Chapter 5 contains concluding remarks and references for further help.

2 Getting Started

2.1 Indigo LabVIEW Toolkit Overview

Five VI libraries are included in the toolkit.

Toolkit Examples.llb contains a series of example applications developed using VIs from the other libraries included in the Indigo LabVIEW Software Toolkit. These can serve to illustrate the use of the toolkit VIs, and in some cases, can serve as starting points for developing custom applications.

Indigo LabVIEW Toolkit Main.llb is the main library containing high-level functions that most software developers will use most often. These include VIs to configure image acquisition and image display, acquire and save images, process images, and so on.

Low Level Toolkit Utilities.llb includes some useful low-level functions that are incorporated into the VIs in **Indigo LabVIEW Toolkit Main.llb** and **Toolkit Examples.llb**.

Thermoteknix Palette Utilities.llb contains VIs designed to facilitate the use of color palettes developed by Thermoteknix Corporation (www.Thermoteknix.com) with images acquired by Indigo Systems cameras.

Extlibsp.llb contains VIs used to embed an IMAQ image in a virtual instrument panel. This library is available at the National Instruments web site, but is included here for convenience.

2.2 Examples

Before we attempt to use the examples we will check to make sure your Indigo Systems camera and National Instruments Framer Grabber Board are operating properly. To do this, navigate to *NI Measurement and Automation Explorer (MAX)*. There is probably a shortcut to MAX on your desktop. Go to *Devices and Interfaces*, and find your IMAQ board. You should see something like the following:

+ IMAQ PCI-1422: img0.

We are after the interface name that follows the colon. If you used the default settings during installation the interface name is probably img0.

Click on the + and you should see something like the following:

Indigo Systems LabVIEW 7 Toolkit Version 1.10

Channel 0: Indigo Systems 320 x 256.

Click on this and then hit the grab button beneath the menu bar. You should see an image from the camera. If no image is acquired then the camera, board, or related software have probably not been properly installed. Please reinstall and try again.

Double-click on the VI library **Toolkit Examples.llb**. The following six top-level VIs will be available: **Example 1.VI**, **Example 2.VI**, **Example 3.VI**, **Example 4.VI**, **Example 5.VI**, and **Camera GUI.VI**. Each of the examples is discussed below. It is assumed that the Indigo Systems camera and National Instruments Frame Grabber boards have been correctly installed.

2.2.1 **Example 1.VI**

Open **Example 1.VI** by double-clicking on the name of the VI in the library. The VI front panel can be minimized, maximized, or closed using the buttons in the upper right corner as usual in Windows applications. This is a very simple example that shows the most basic use of some of the VIs in the toolkit.



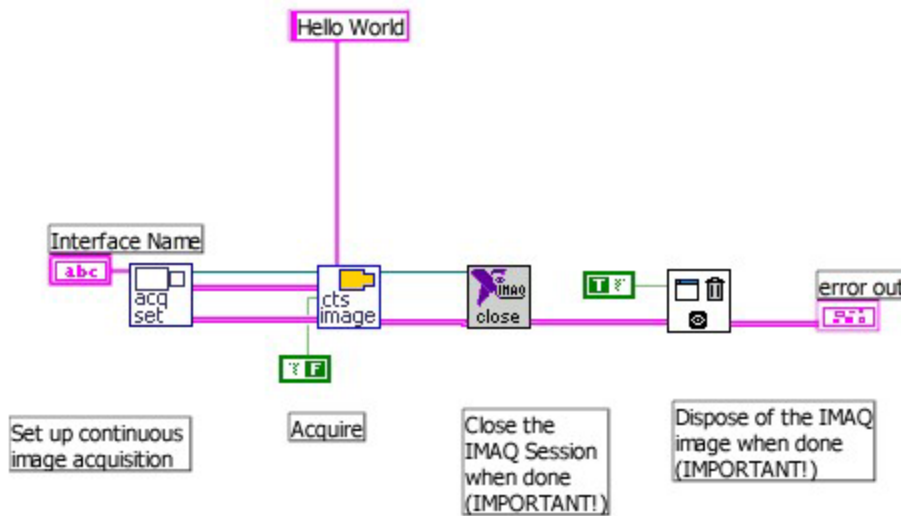
Running Example 1.VI

Type the name of your interface into the *interface name* control and hit the run button. You should see another VI front panel pop up and you should see an image pop up in a separate window. Note that the image window can hide behind other windows on your desktop.

Wave your hand in front of the camera to verify continuous acquisition of images and then hit the **stop** button on the pop-up VI.

Let's now look at the diagram for **Example 1.VI**.

Indigo Systems LabVIEW 7 Toolkit Version 1.10



The first VI in the diagram is  **Image Acquisition Setup.VI**. This sets up the image acquisition parameters such as interface name, image name, memory location (on board or on computer), edge removal selection, and edge size for removal. Most of the inputs are unconnected in this example because the defaults are satisfactory. The most important output of this VI is the IMAQ session.

The IMAQ session must be connected to the next VI in the chain,  **Continuous Image Acquire.VI**. Double-click on the VI's icon to open the front panel. Notice that this is the pop-up VI that appeared when **Example 1.VI** was run. Right-click on the icon and verify under VI properties that the window display has been customized to include the pop-up feature. Continuous Image Acquire.VI has as inputs the IMAQ session that must be connected, window number for display (defaults to IMAQ window zero), a color palette input that allows false color schemes to be applied to the image, and an image title that appeared at the top of the IMAQ image.

The IMAQ session out of **Continuous Image Acquire.VI** must be connected to the next VI, the **IMAQ Close Session.VI**. It is essential that all IMAQ programs close their IMAQ sessions at the end of a run. Failing to do this will likely result in a system crash. Finally, all IMAQ windows are closed. It is often advisable to set the VI properties of an application to hide the abort button and force the user to stop the application via a front panel button, thus allowing the programmer to terminate execution in a controlled manner. To access this capability, right-click on the VI icon and select *VI Properties*. In the pop-up window select *Window Appearance* and then *Customize*. Deselect *Show Abort Button*. This technique is employed in this and other examples included in the Toolkit.

Notice that error propagation is included in this example. VIs in this toolkit include an error input cluster and an error output cluster. Use of the error propagation and error handling features in LabVIEW is strongly encouraged.

Each of the Indigo LabVIEW Toolkit VIs in this example and the following examples are described more fully in chapter 3.

2.2.2 Example 2.VI

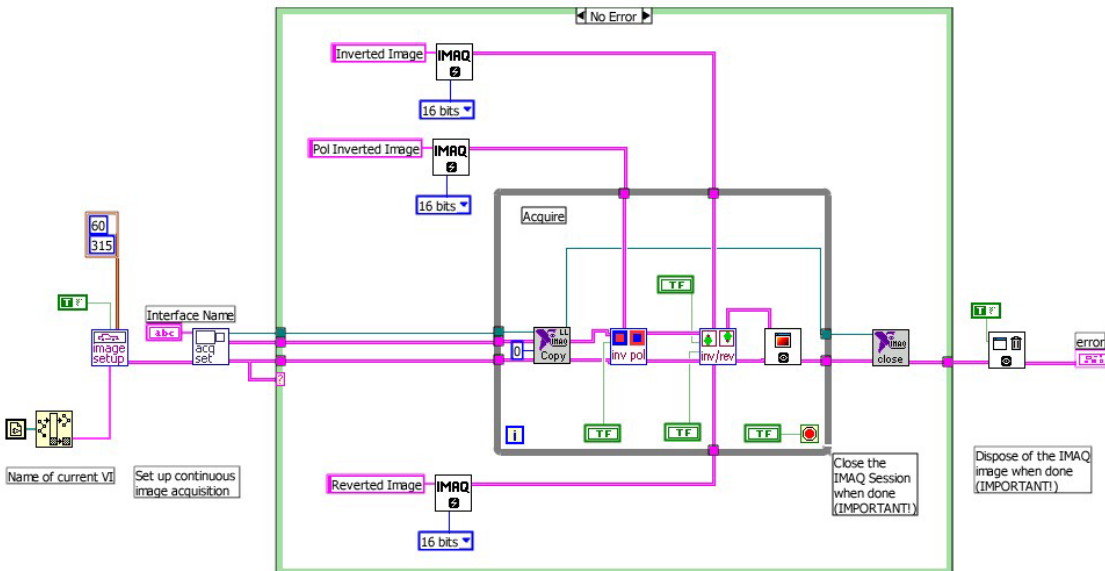
Minimize or close **Example 1.VI** and open **Example 2.VI**.



Type your interface name into the control and hit run. Notice that the image now appears to be built into the VI panel. We say that it is *embedded* in the front panel of **Example 2.VI**. Hit the polarity, revert, and invert buttons and observe the results, ideally while waving your hand in front of the camera. Hit the stop button.

Indigo Systems LabVIEW 7 Toolkit Version 1.10

Let's look at the diagram for **Example 2.VI**.



Note that we now begin with the VI  **Image Display Setup.VI**. This VI sets up the image display according to the user's needs. Hoover over the icon and observe the various connections. In this case we have elected to embed the image in a VI front panel. To do this we must specify the *exact* name of the VI and the coordinates where we would like to embed the image. Since the image is to be embedded in the panel of the VI that calls **Image Display Setup.VI**, we have used the standard LabVIEW functions **current VI's path constant** and **strip path** to obtain the name of the calling VI. Save a copy of **Example 2.VI** in a convenient location. Try changing the coordinates of the embedded image and run the VI again. Be sure not to save any changes to the original. While in edit mode, note that the VI panel will still scroll under the embedded image. It is often advisable to set the VI properties of an application to disable scrolling when an image is embedded. This will prevent users who are inexperienced with LabVIEW programming from inadvertently scrolling important parts of the front panel under the image or out of the visible range while running. Usually users will not have access to edit mode since applications are typically distributed as executables. To disable scrolling, right-click on the VI icon and select *VI Properties*. In the pop-up window select *Window Appearance* and then *Customize*. Deselect *Show Scroll Bars*. This technique is employed in this and other examples included in the Toolkit.

We next use the  **Image Acquisition Setup.VI** as in **Example 1.VI**. If no error is detected we then begin to acquire and display images. Note the while loop that copies an image per iteration from the buffer and then passes it to  **Polarity.VI** and then to  **Invert_Revert.VI**. Notice that each of these VIs accepts an input image and a reference to the altered image, for example,

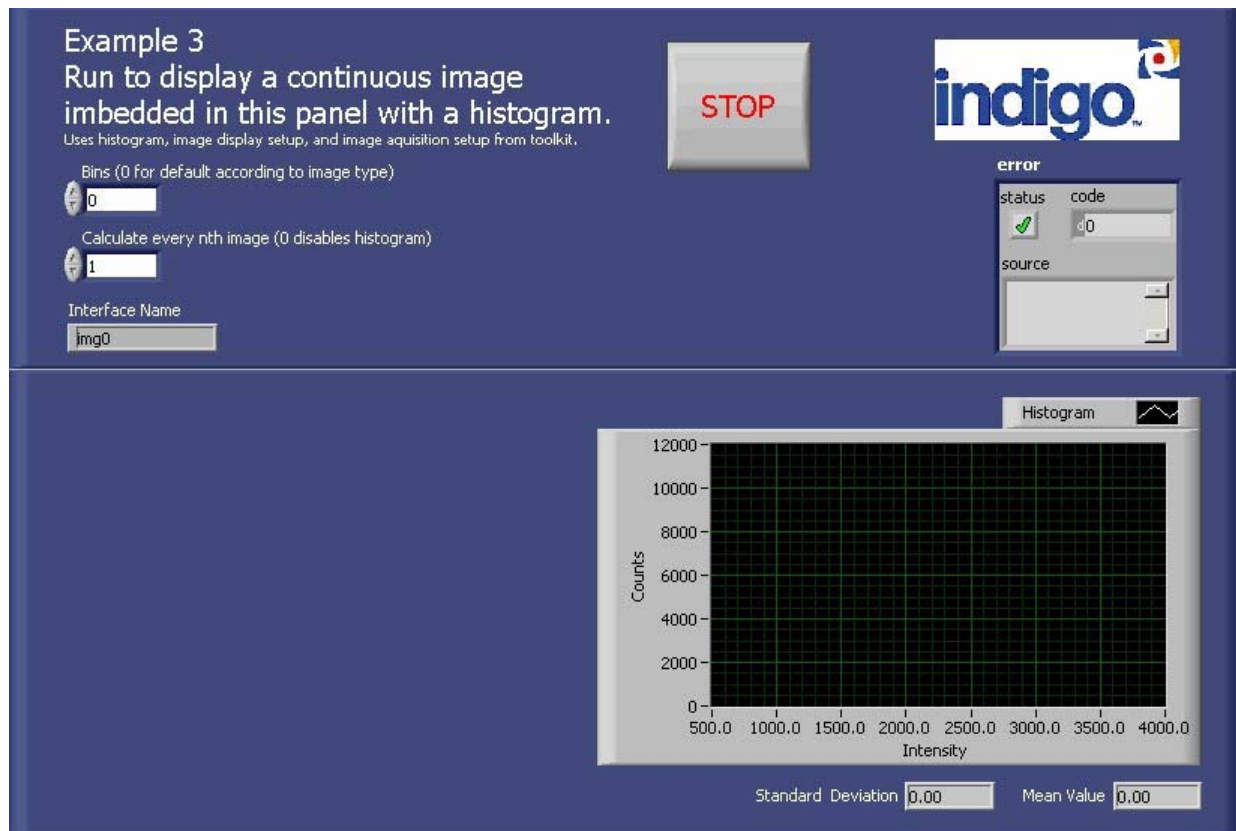
Indigo Systems LabVIEW 7 Toolkit Version 1.10

polarity inverted image in **polarity.VI**. In **Example 2.VI** we have chosen to wire the reference inputs to newly created IMAQ images with different names. As an alternative we could have wired the input image to both the image input and the altered image reference. But this would cause the polarity inverted image for example to overwrite the original with permanent loss of the original image. This can cause a variety of bad outcomes. In your saved copy of **Example 2.VI** disconnect the reference to the newly created Pol Inverted Image and connect the image from the **IMAQ copy** function to that input as well as the image input. Run the VI and observe the results. Restore the original diagram.

The altered image is finally displayed per iteration using the **IMAQ WindDraw** function. The loop acquires and displays continuously until the stop button on the front panel is pressed, and then the IMAQ session is closed and images are discarded.

2.2.3 Example 3.VI

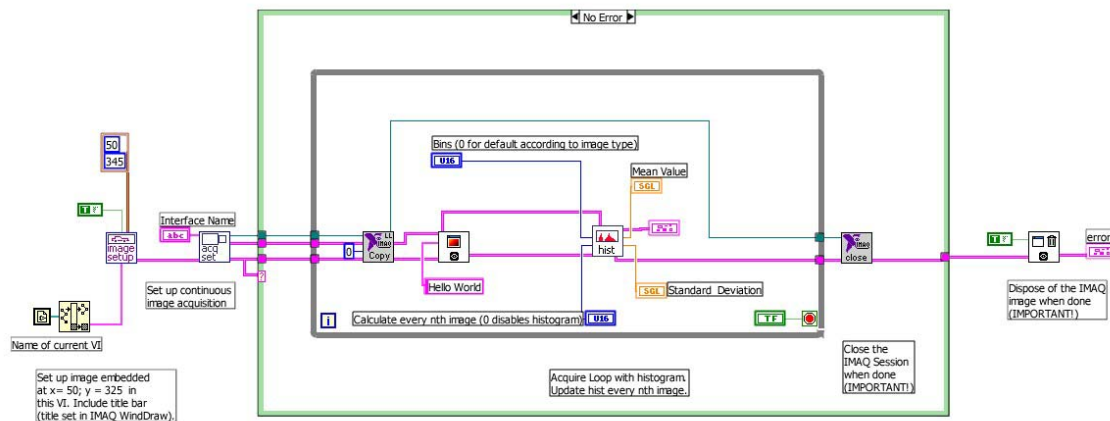
Minimize or close **Example 2.VI** and open **Example 3.VI**.



Indigo Systems LabVIEW 7 Toolkit Version 1.10

Enter your interface name and hit run. Notice that the embedded image is again displayed as in **Example 2.VI**, but now there is also a histogram displayed to the right of the image. Set the histogram update period (calculate every nth image control) to 0 and notice that the histogram is disabled. Set this control to 10 and notice that the image updates rapidly while the histogram updates only every tenth image. Adjust the number of bins in the histogram and observe the results. Stop the VI and switch to the diagram.

Let's look at the diagram for **Example 3.VI**.



Notice that we again begin with  **Image Display Setup.VI** and then continue to  **Image Acquisition Setup.VI** as in **Example 2.VI**. The acquisition loop is also set up in a similar manner, but the  **Histogram.VI** is new. This VI calculates the histogram of all the image pixels. It can be applied to 8 bit or 12 bit images. The number of bins in the histogram can be set by the user, but the default setting of zero will result in automatic adjustment of the number of bins in the histogram to the image resolution (256 for 8 bit images, 4096 for 12 bit images).

2.2.4 Ex.3 Example 3.VI Modified

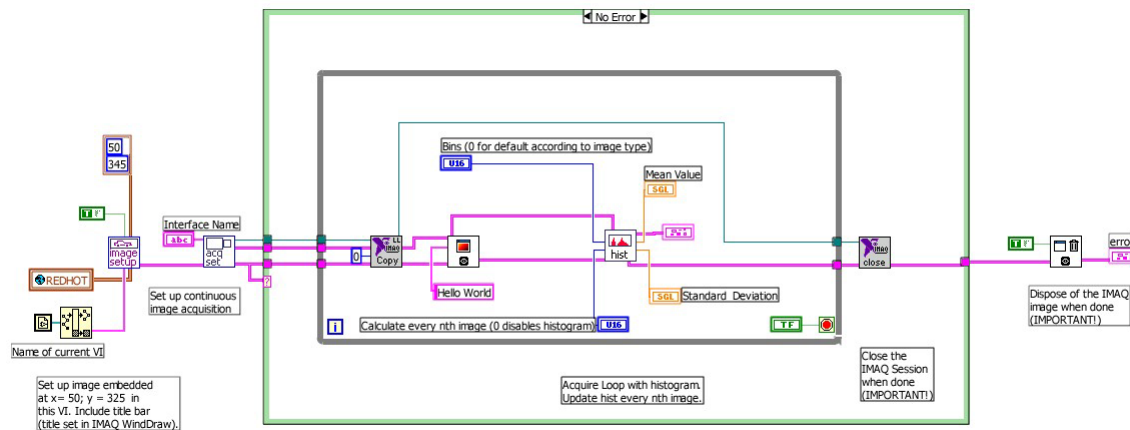
Color palettes apply false color schemes to 8 bit images. For example, a color palette might assign the color bright red to a very high intensity pixel while assigning the color dark blue to a very low intensity pixel. Many color palettes have been developed for a variety of different purposes, often allowing important features in an image to be enhanced.

The Indigo LabVIEW Toolkit provides support for two sets of palettes. The global variable  **Indigo Palettes Global.VI** contains a number of standard color palettes provided by Indigo Systems Corporation, and these can be connected directly to the Color Palette inputs on a number of VIs in the toolkit. In addition the Toolkit supports the use of popular color palettes from Thermoteknix

Indigo Systems LabVIEW 7 Toolkit Version 1.10

Corporation. The use of Thermoteknix palettes is described in **Example 4.VI** below. For now we will use the Indigo palettes.

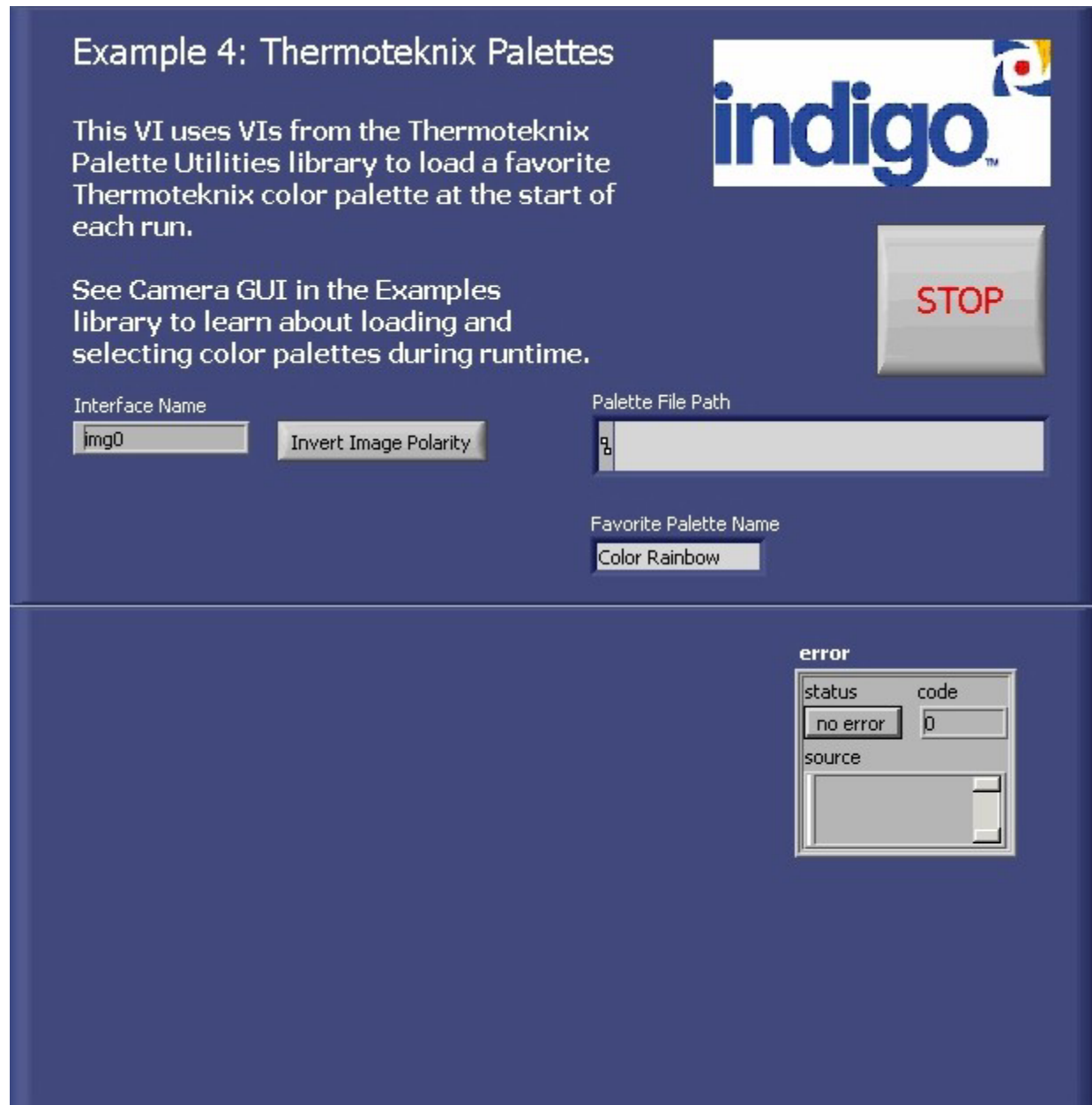
Open the diagram for **Example 3.VI** and save it under a different name. Insert  **Indigo Palettes Global.VI**. This VI is in the Low Level Toolkit Utilities library. Select the REDHOT palette and change the global to a read global. Connect to the palette input. The diagram should look like the one below.



Run the VI and observe the image. Stop the VI, select another color palette and run again. **Camera GUI.VI** shows a way to change the color palette dynamically.

2.2.5 **Example 4.VI**

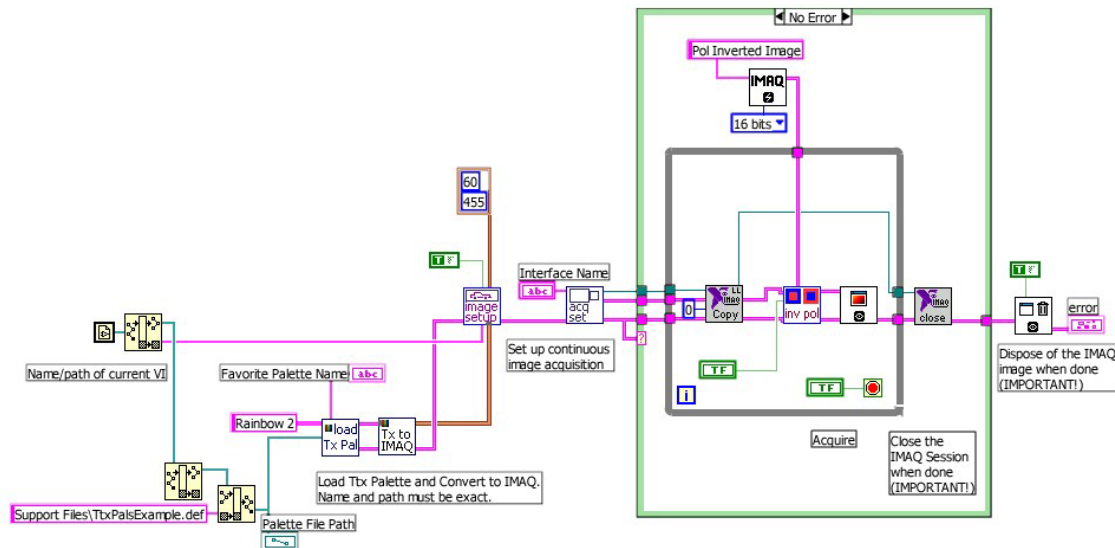
Minimize or close **Example 3.VI** and open **Example 4.VI**.



This example is designed to illustrate the use of Thermoteknix Color Palettes in Indigo camera applications. Support for use of these palettes is extensive and this example illustrates only a simple application of these popular palettes.

Let's look at the diagram for **Example 4.VI**.

Indigo Systems LabVIEW 7 Toolkit Version 1.10



Most of the example is similar to **Example 2.VI** and **Example 3.VI**, but notice that the color palette

connection for  **Image Display Setup.VI** is now connected to a series of VIs associated with Thermoteknix Color Palettes. These VIs can be found in the Thermoteknix Palette Utilities library. In this example we load a favorite Thermoteknix palette. To load a particular palette we can use



Load Thermoteknix Color Palette by Name.VI which requires as inputs the name of the palette and the file path to the color palette file that contains this palette. Thermoteknix palette files have the extension .def. The structure of the files is discussed later in the Toolkit Programmer's Manual. The Thermoteknix palette is then converted to an IMAQ color palette using  **Construct IMAQ Palette.VI**.

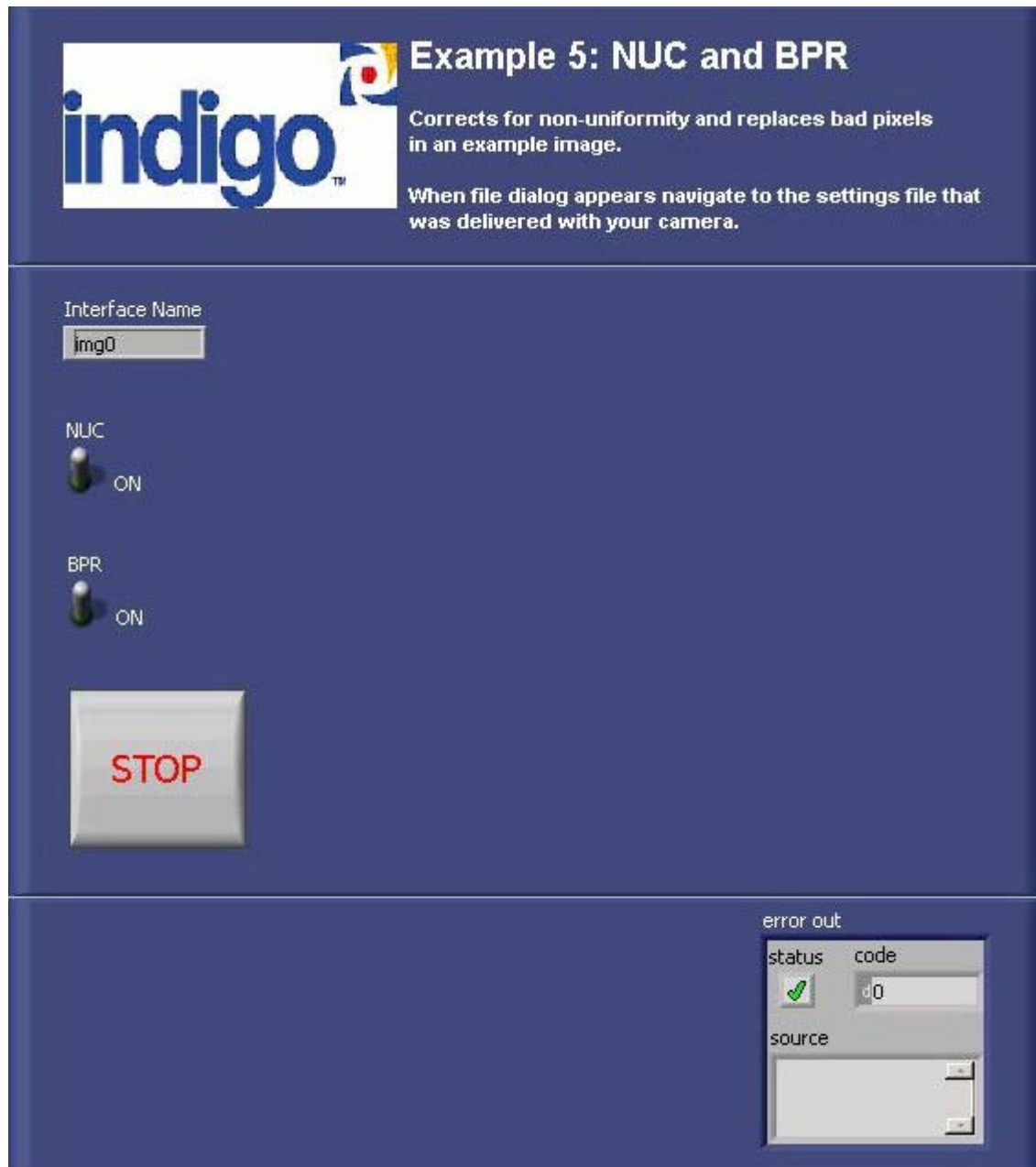
Other VIs in the Thermoteknix Palette Utilities library open a .def file and find the names of all the

palettes in the file.  **Load Thermoteknix Palette GUI.VI** provides a simple graphical user interface to facilitate working with these palettes. In addition, a function to convert IMAQ palettes to LabVIEW color tables is included so that IMAQ and/or Thermoteknix color palettes can be used with LabVIEW picture controls.

2.2.6 **Example 5.VI**

Minimize or close **Example 4.VI** and open **Example 5.VI**.

**Indigo Systems LabVIEW 7 Toolkit
Version 1.10**



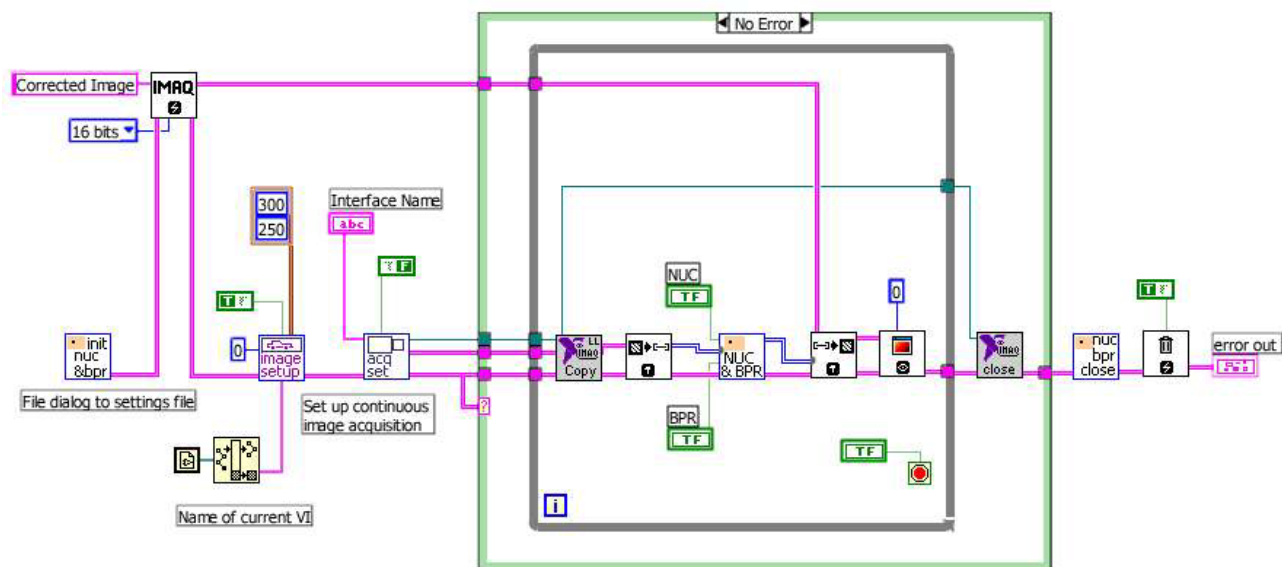
This example illustrates the use of the *Non-Uniformity Correction (NUC)* and *Bad Pixel Replacement (BPR)* routines included in the toolkit library **NUC and BPR.llb**. Readers unfamiliar with NUC and BPR may wish to take a detour to section 3.3 where the algorithms are discussed.

If external NUC and BPR are appropriate for your Indigo Systems camera a *settings file* will have been provided with the camera. Locate this file and run **Example 5.VI**. A file dialog will ask you to navigate to the settings file. An image will then be displayed, imbedded as usual in the panel. If convenient, point the camera at a fairly uniform surface so that the image appears as a relatively

Indigo Systems LabVIEW 7 Toolkit Version 1.10

uniform shade of gray. Switch off BPR using the front panel control. Careful examination of the image should reveal a small number of bad pixels that will most likely appear as black or white specs, often near the edge of the image. If no bad pixels appear do not be concerned. It is possible that your camera has absolutely no bad pixels. Now turn off NUC. You should see some variations in the image, perhaps in the form of vertical or horizontal striations. This reflects the non-uniformity of the gains and offsets of individual pixels in the focal plane. Turn NUC and BPR on and notice that the image returns to its original quality.

Let's look at the diagram for **Example 5.VI**.



As in previous examples we set up image display and image acquisition and then acquire continuously in a loop until the user hits the stop button.

However, before attempting to set up image display and image acquisition we first use

init nuc & bpr **Initialize NUC and BPR.VI** to initialize the NUC and BPR dll. This initialization VI accepts as an input a file path to the settings file, but since we did not connect this input a file dialog resulted. Note that the initialization function in this dll **MUST** be called before other functions in the dll are called. Failure to initialize the dll may result in a system crash. The dll should not be initialized per frame as this will slow acquisition.

In the acquisition loop the IMAQ image is converted to an array and this array is passed to **NUC & BPR** **NUC and BPR.VI**. The array of corrected image pixels is then converted back to an IMAQ image for display.

Caution

Indigo Systems LabVIEW 7 Toolkit Version 1.10

The NUC and BPR routine uses data from the settings file that include the gain and offset of each pixel in the focal plane, as well as a bad pixel map. Initialization of the dll involves passing these tables of data from LabVIEW to the dll. It is essential that the correct settings file for the particular camera (serial number specific) be used. Also, it is essential that the entire image be passed to both



Initialize NUC and BPR.VI and  **NUC and BPR.VI**. An attempt to use these VIs on an image of a different size will likely result in a system crash. In particular, any ROI selection should take place after NUC and BPR have been completed.

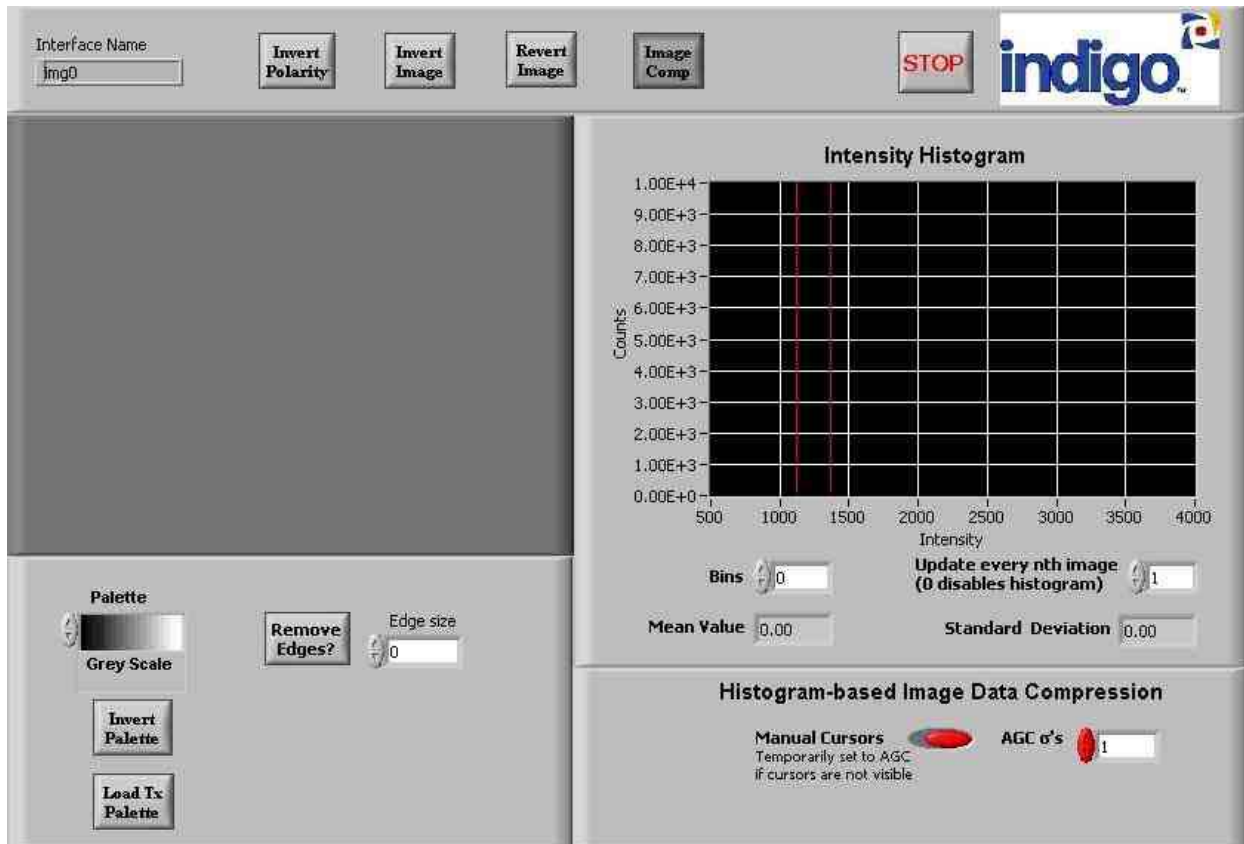
Using the lower level VIs in **NUC and BPR.llb**, it is possible to get around this limitation and apply NUC and BPR routines directly to an ROI rather than selecting the ROI from a corrected image. Applying NUC and BPR to the ROI has the advantage of increasing video throughput since fewer calculations are required, but considerably more coding is needed to implement NUC and BPR on ROIs. Programmers interested in pursuing this may wish to contact Indigo Systems Corporation for guidance. The first pages of the manual contain contact information.

After the user exits from the acquisition loop the IMAQ session is closed, the NUC and BPR library is closed using  **Close NUC_BPR Library.VI**, and IMAQ images are destroyed using IMAQ Dispose.VI. All three of these steps are important.

2.2.7 Camera GUI.VI

Minimize or close **Example 5.VI** and open **Camera GUI.VI**.

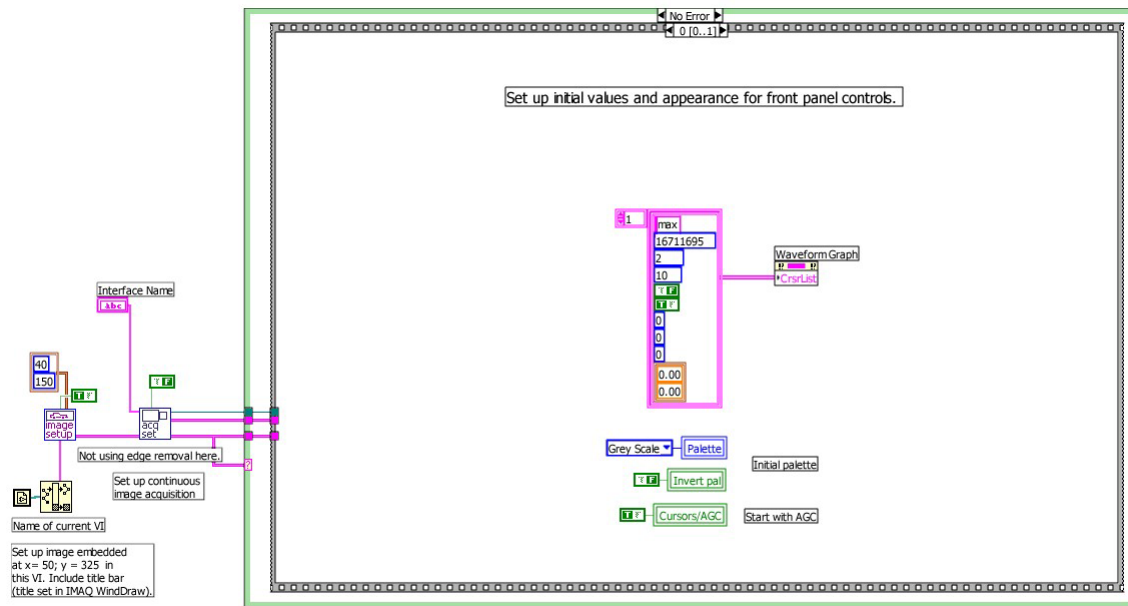
Indigo Systems LabVIEW 7 Toolkit Version 1.10



This example is much more complicated than the others. It is designed to illustrate the use of the Toolkit VIs along with IMAQ and LabVIEW functions to produce a complete application.

Let's look at the diagram for **Camera GUI.VI**.

Indigo Systems LabVIEW 7 Toolkit Version 1.10

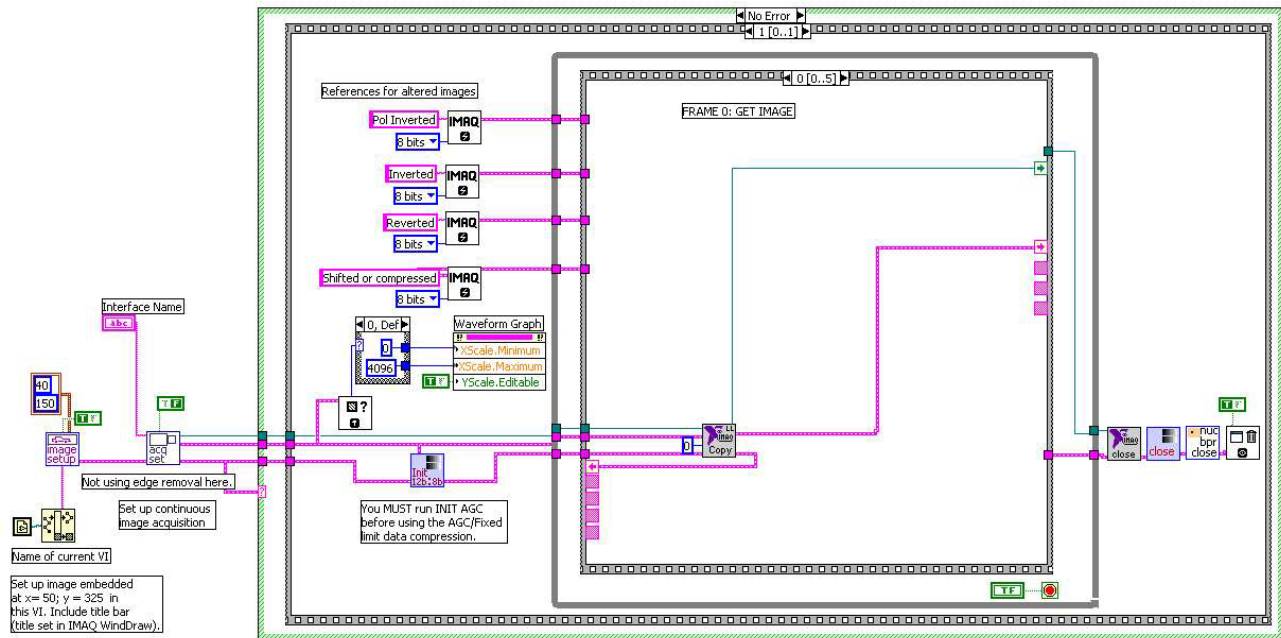


As in the other examples, we begin by configuring image display and image acquisition using

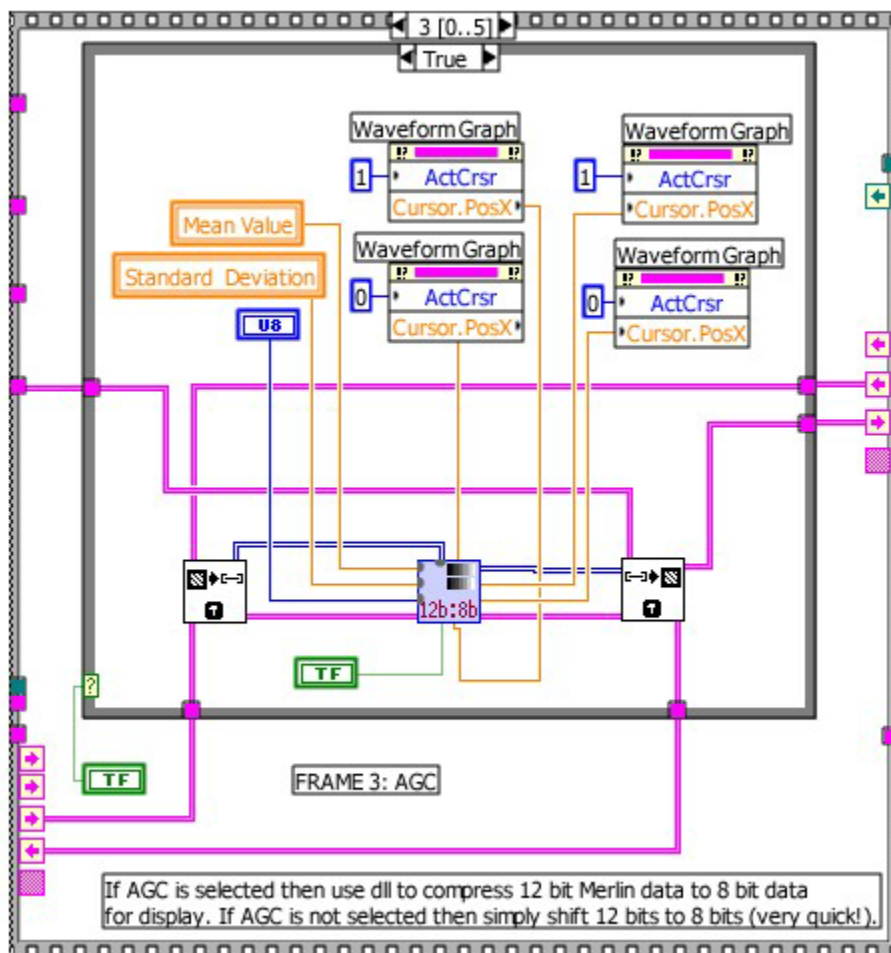
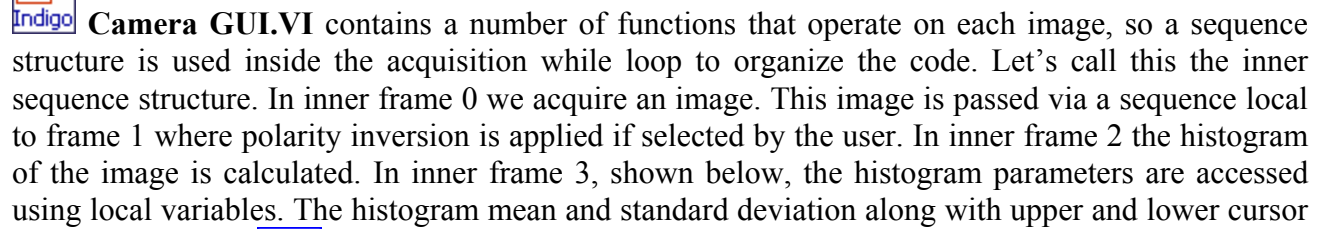


Image Display Setup.VI and then continue to  **Image Acquisition Setup.VI**. Provided no error is generated during setup, we enter the sequence structure frame 0. In this frame we initialize some variables and the appearance of the cursors on the histogram.

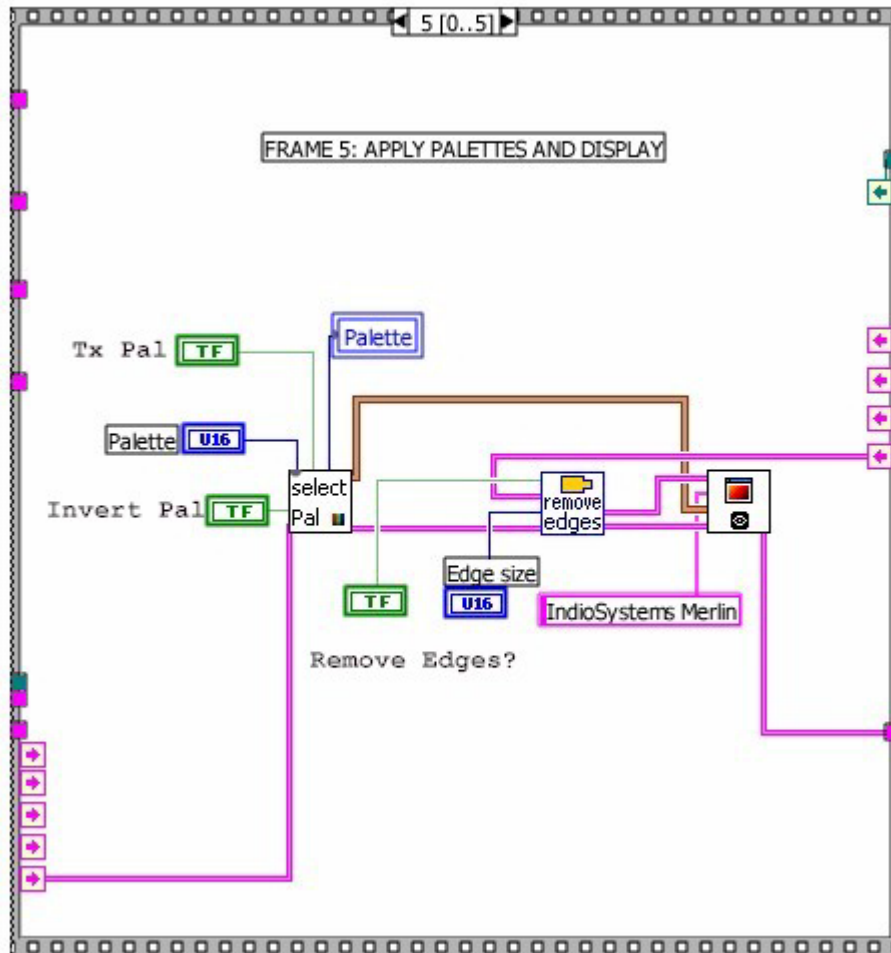
Indigo Systems LabVIEW 7 Toolkit Version 1.10



The next frame of the sequence structure contains the main acquisition loop (the while loop). Before the main loop we initialize Automatic Gain Control using  **Image Compression Init.VI** Image intensity compression, or windowing, is discussed later in the manual. Users unfamiliar with image intensity windowing may wish to take a detour to the discussion of  **Image Compression Init.VI** and  **Image Compression.VI** in sections 3.14-3.15 before proceeding.



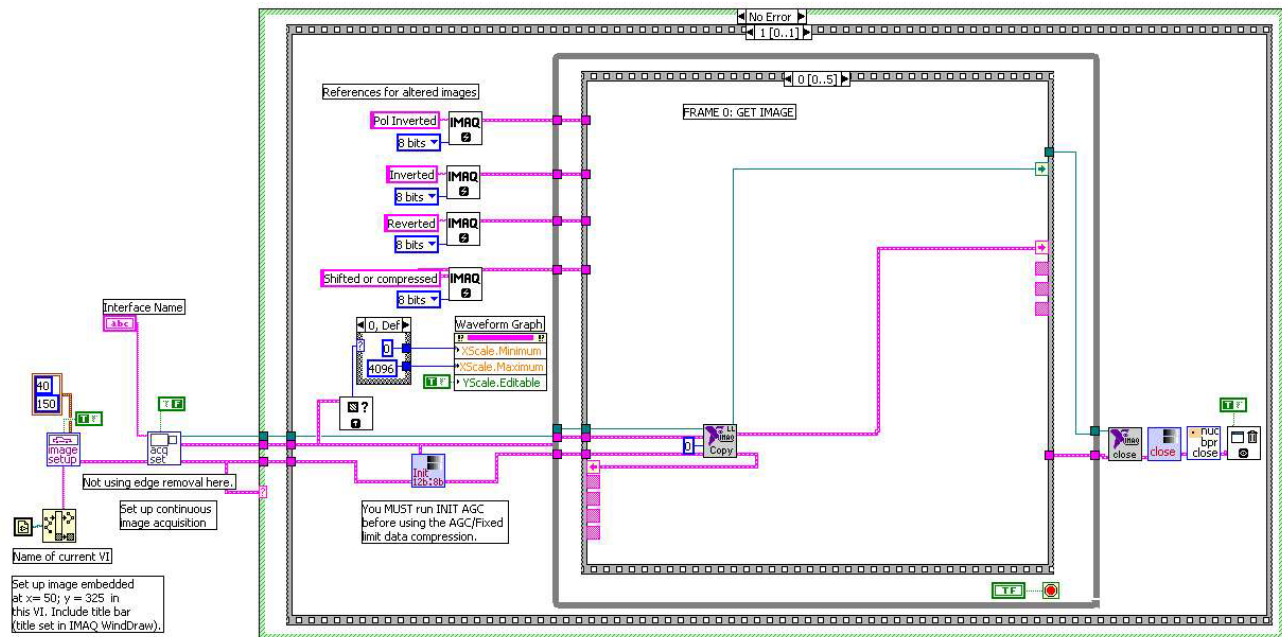
In inner frame 5 the color palette is selected as shown below.



Notice that this is a more sophisticated use of color palettes than was illustrated in the previous examples. In **Camera GUI.VI** the user can change color palettes during operation using  **Set Color Palette.VI**. Operate the associated buttons and observe the operations.

Consider again the diagram for Camera GUI.VI:

Indigo Systems LabVIEW 7 Toolkit Version 1.10



After the user exits from the acquisition while loop the IMAQ session is closed, the image compression dll is closed using  Close Toolkit Library.VI, the NUC and BPR dll is closed using  Close NUC_BPR Library.VI, and all IMAQ images are destroyed using IMAQ Dispose.VI.

3 Toolkit VI Reference

This chapter contains an overview of each VI in the toolkit libraries **Indigo LabVIEW Toolkit Main.llb**, **Thermoteknix Palette Utilities.llb**, and **Low Level Toolkit Utilities.llb**. One additional library, **Extlibsp.llb**, available at the National Instruments web site, is included here for convenience. The library **Toolkit Examples.llb** is described in chapter 2. It is recommended that the reader examine the use of the VIs described below in the examples before attempting to use them in custom applications.

The icon, name, and front panel of each VI is presented below, followed by a description and a list of inputs and outputs. An *asterisk on an input indicates that it is required for the VI to run, and parenthesis enclose defaults and brief comments about inputs and outputs where appropriate.

3.1 Indigo LabVIEW Toolkit Main.llb

3.1.1 Close Toolkit Library.VI



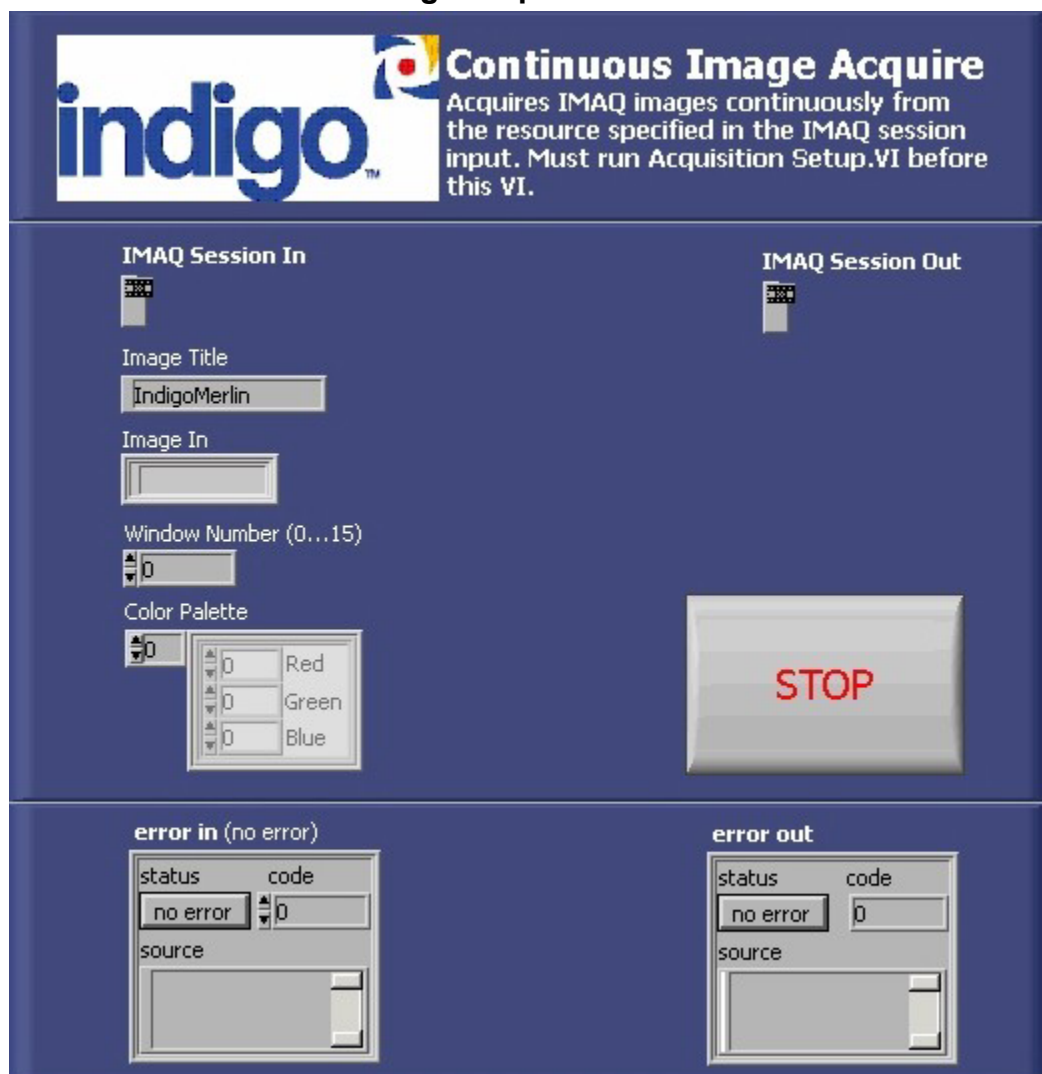
This VI closes the dll used for image data compression. This VI should be run once as the application closes if  **Image Compression.VI** and  **Image Compression Init.VI** are used.

Indigo Systems LabVIEW 7 Toolkit Version 1.10

See the discussion of  **Camera GUI.VI** in Chapter 2 for an example of the use of image compression VIs.

Inputs: Error Cluster
Outputs: Error Cluster

3.1.2 Continuous Image Acquire.VI



This VI initiates continuous image acquisition to the specified image window.  **Image acquisition setup.VI** must have been run before this VI is called and the IMAQ handle generated

Indigo Systems LabVIEW 7 Toolkit
Version 1.10

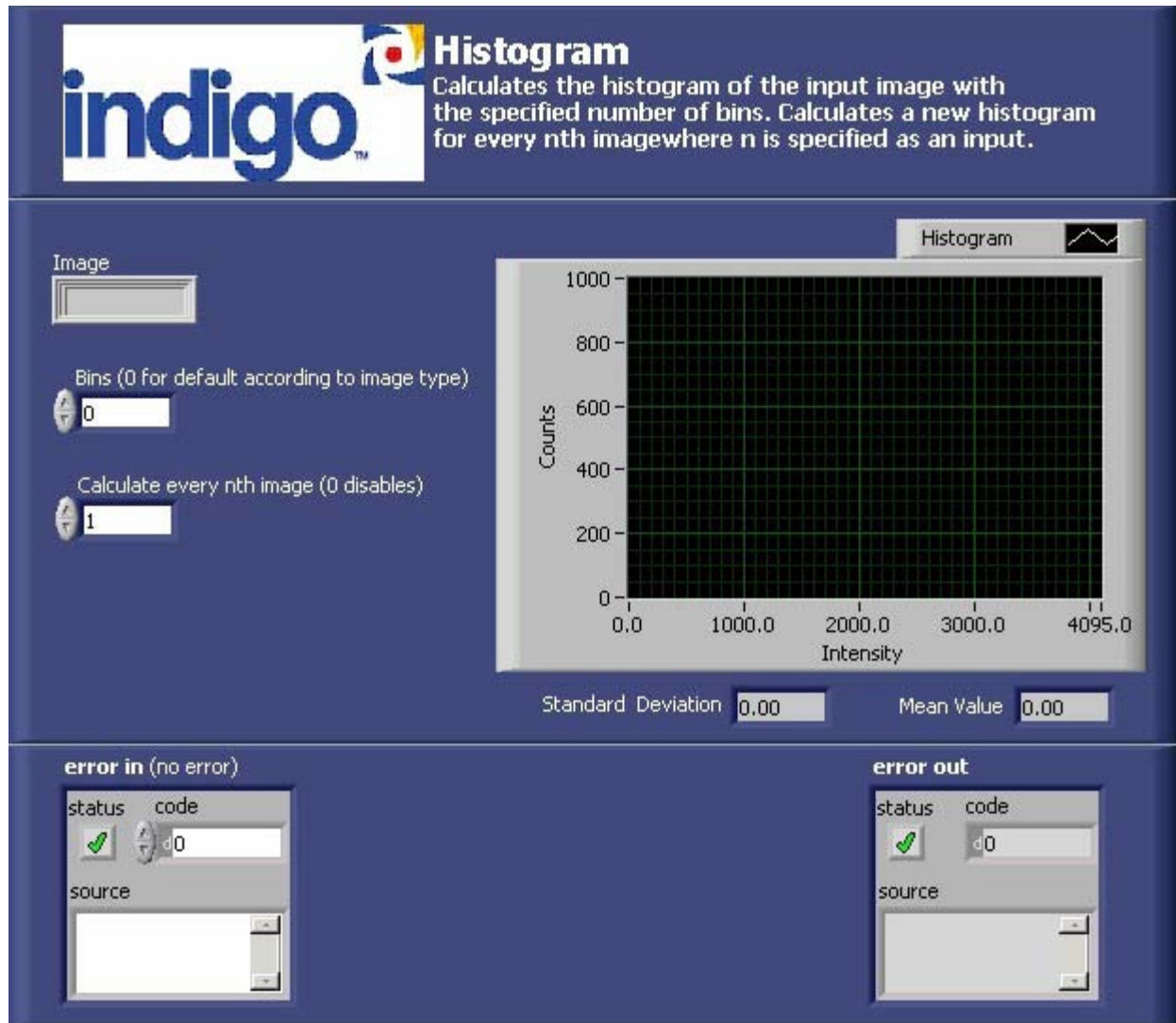


there must be passed to this VI. If **Image display Setup.VI** has been run, the image will be displayed as per settings. If not, IMAQ default settings will be used to display the image.

This VI will pop up when called and will acquire and display images continuously until the stop button is pressed.

Inputs:	IMAQ session in*
	Image Title (IndigoImage)
	Image In* (reference to image to be acquired)
	IMAQ image window number (0 – 15, defaults to 0)
	Color Palette
	Error Cluster
Outputs:	IMAQ session out
	Error Cluster

3.1.3 Histogram.VI



This VI calculates the histogram of an IMAQ image.  **Histogram.VI** works on either 8 bit or 12 bit images. The user can specify the number of bins in the histogram, but if this is not connected or is set to zero the histogram will default to a number of bins equal to the possible intensities in the image. For example, the histogram for a 12 bit image would default to 4096 bins. The user can specify the frequency with which the histogram is updated. If this is set to 1 the histogram will update for each image. If it is set to zero the histogram will not update. Since histogram calculations are time consuming, programmers often choose to update the histogram perhaps every tenth image. The mean and standard deviation outputs are for the last image for which a histogram was calculated.

Indigo Systems LabVIEW 7 Toolkit Version 1.10

Inputs: Image* (reference to IMAQ image)
 Number of bins in histogram (see discussion of defaults above)
 Calculate histogram for every nth image (1)
 Error Cluster

Outputs: Histogram cluster
 Mean
 Standard Deviation
 Error Cluster

3.1.4 Image Acquisition Setup.VI



Image Acquisition Setup
Sets up image acquisition. Interface name, image name, and memory location are required inputs.

Interface Name:

Image Name:

Memory Location:

Remove Edges? ☐ Edge size:

ROI (for edge removal):

error in (no error)

status: ☒ code:

source:

error out

status: ☒ code:

source:

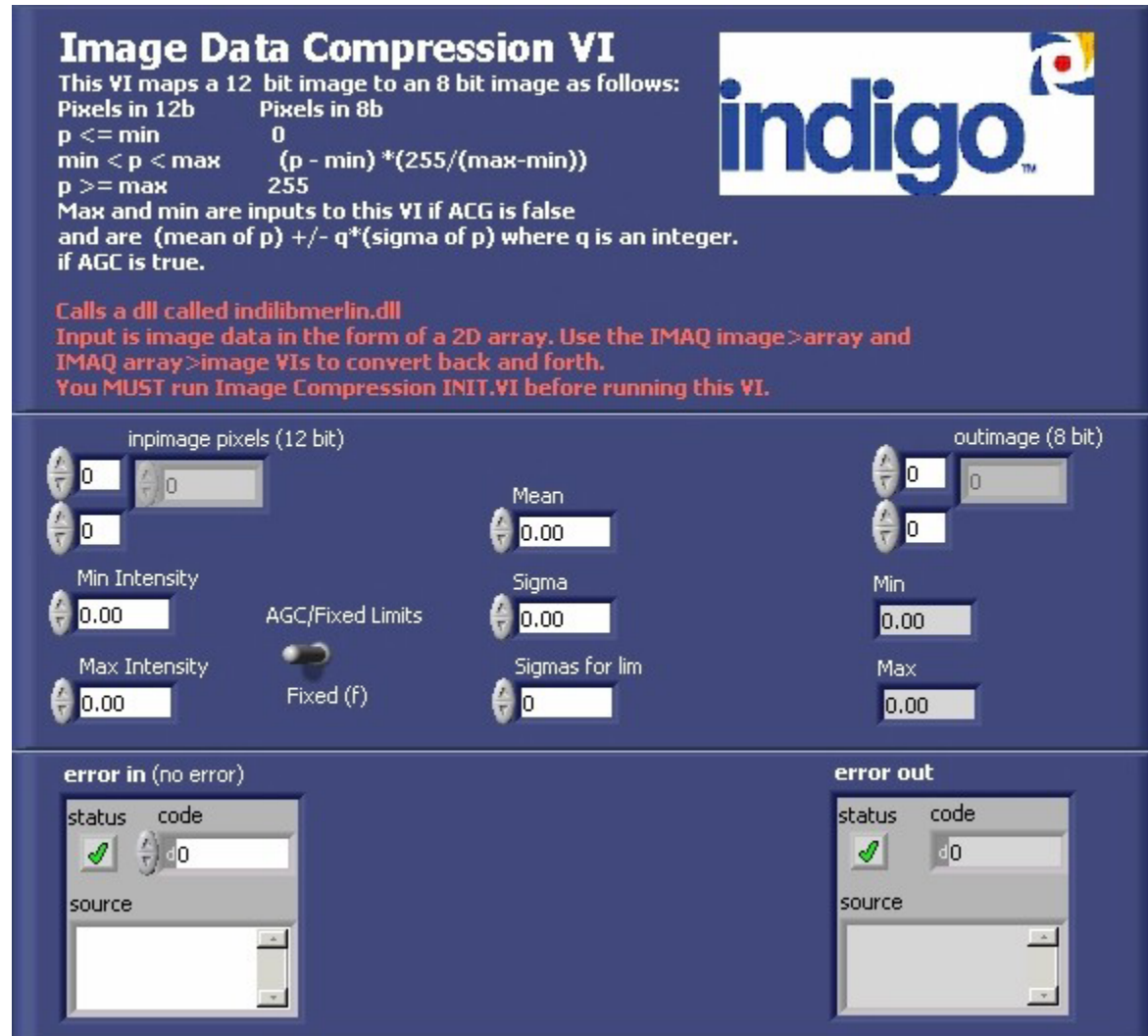
This VI sets up image acquisition parameters. The interface name must be exactly correct and can be obtained for your system using National Instruments Automation Explorer. The image name is optional and defaults to IndigoImage. The memory location to which images are written can be either the system (computer RAM) or the RAM on the IMAQ board. Often each new image will overwrite the previous image, and thus memory is not an issue. However, if multiple images or

**Indigo Systems LabVIEW 7 Toolkit
Version 1.10**

image sequences are to be acquired, use of RAM can be considerable. Consult documentation for your system and board to determine the amount of memory available in each case. If edge removal is selected the specified number of pixels will be removed from each edge of the image. This is often employed to remove bad pixels that are frequently found near the edge of the focal plane.

Inputs:	Interface Name* (often img0, see MAX)
	Image Name (IndigoImage)
	Memory location (System)
	Edge Removal Select (f)
	Edge size for edge removal (0, ignored if edge removal is not selected).
	Error Cluster
Outputs:	IMAQ Session
	Image (reference to IMAQ image)
	Region of Interest for edge removal
	Error Cluster

3.1.5 Image Compression.VI



This VI compresses a twelve bit image to an eight bit image. The compression proceeds as follows:

Pixels in 12 b image	Pixels in 8 b image
$P_{ij} \leq I_{\min}$	0
$I_{\min} < P_{ij} < I_{\max}$	$(P_{ij} - I_{\min}) * (255 / (I_{\max} - I_{\min}))$
$P_{ij} \geq I_{\max}$	255

$I_{\max}$ and $I_{\min}$ are inputs to this VI if AGC/Fixed Limits is set to Fixed Limits. If automatic gain control (AGC) is selected, $I_{\max}$ and $I_{\min}$ are calculated from the histogram data as follows:

$$I_{\max} = \text{mean} + (\text{sigmas for lim}) * \text{sigma}$$

Indigo Systems LabVIEW 7 Toolkit
Version 1.10

$$I_{\min} = \text{mean} - (\text{sigmas for lim}) * \text{sigma}$$

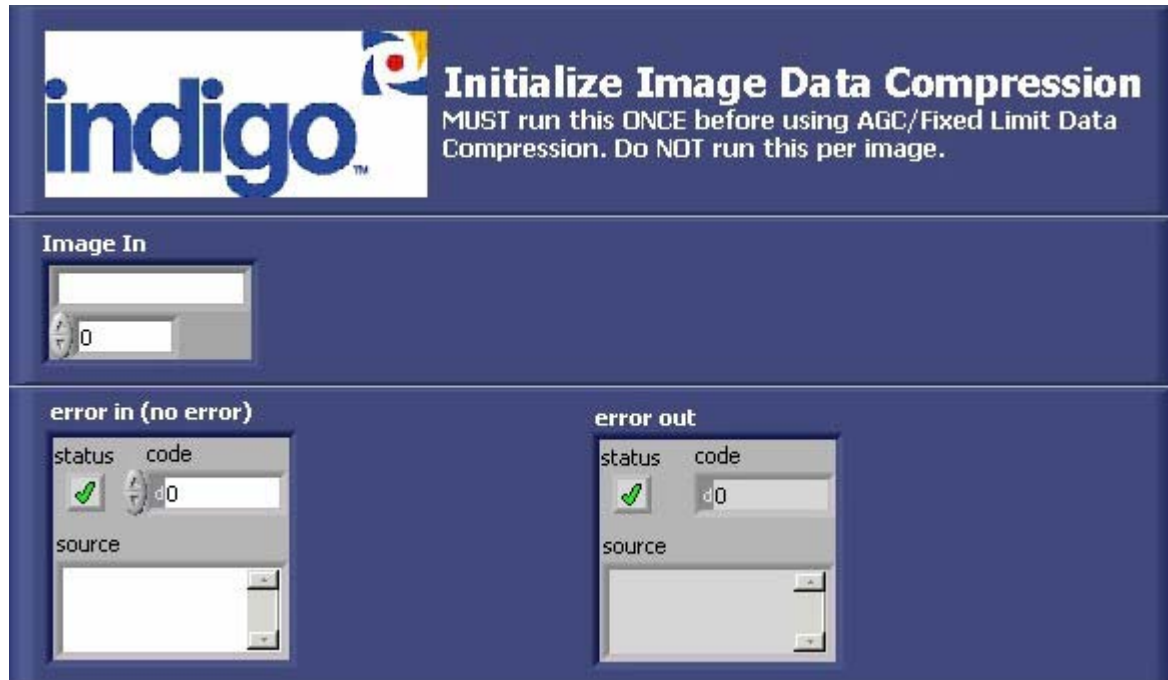
The image data is fed into this VI in the form of a 2D integer array (I16 data type) Use the IMAQ image to array and array to image VIs to switch back and forth as needed. This is illustrated in Camera GUI.VI in the Toolkit Examples library.

The outputs of this VI are the 8 bit image pixels (U8 data type) and the minimum and maximum intensity used in the compression routine. If for any reason the minimum intensity is fed into this VI or calculated from the histogram data is less than zero then zero will be used as the minimum intensity in the compression routine. Similarly, if the maximum intensity fed into this VI or calculated from the histogram data is larger than 4095 then 4095 will be used as the maximum intensity in the compression routine.

It is essential that  **Image Compression Init.VI** be run before this VI is run. If this is not done a system crash is likely. But the Init VI should not be run each time an image is compressed. See instructions for the Init VI. Also,  **Close Toolkit Library.VI** should be run once just before the application that used image compression closes.

Inputs:	*Inpimage (the input image pixels array, I16 data type) Maximum Intensity (0, ignored if AGC is selected) Minimum Intensity (0, ignored if AGC is selected) Fixed Limits / AGC (fixed limits) Mean (0, ignored if fixed limits is selected) Sigma (0, ignored if fixed limits is selected) Sigmas for limit (0, ignored if fixed limits is selected) Error Cluster
Outputs:	Image pixels out (U8) Minimum Intensity employed in intensity windowing Maximum Intensity employed in intensity windowing Error Cluster

3.1.6 Image Compression Init.VI



This VI initializes the dll used by  **image compression.VI**. It is essential that the initialization VI execute before the compression VI. If the Init routine is not run first the likely result is a system crash and loss of any unsaved LabVIEW code. The Init VI should not be run for each image as this will be too time-consuming. However, if the image changes size for any reason (*e.g.*, edge removal is turned on dynamically or a new ROI is applied), then the Init VI must be run again.

Inputs: Image* (Reference to IMAQ image of size and type to be compressed)
 Error Cluster

Outputs: Error Cluster

3.1.7 Image Display Setup.VI



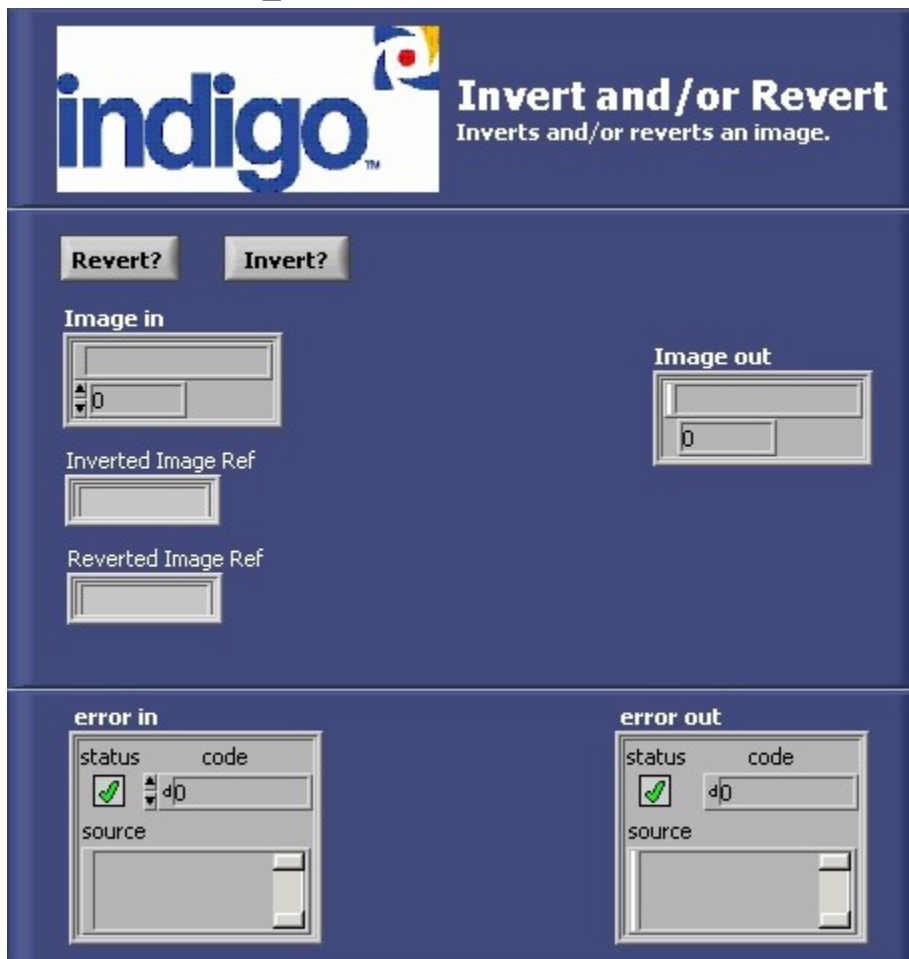
This VI formats the display of the IMAQ image destined for the specified IMAQ window. The zoom factor will zoom into or out of an image, making the window up to 16 times larger (with a zoom factor of positive 16) or sixteen times smaller (with a zoom factor of negative 16). The title bar can be displayed or not according to the title bar selector, and the image can be set to visible or hidden. By default IMAQ images are displayed in a separate window that can be dragged to any desired location on the screen, closed, and minimized. Even if the VI that produced the image is closed, the IMAQ image will remain. Often it is better to embed an image in a LabVIEW panel, making it part of the panel. When this is done the IMAQ image will move when the VI panel window is moved, but the panel will still scroll under the image. When embedding is selected the name of the panel must be specified exactly. The coordinates of the upper left corner of the image can also be specified, along with an optional color palette. The coordinates and the color palette can be updated dynamically.

**Indigo Systems LabVIEW 7 Toolkit
Version 1.10**

Inputs: Image
 IMAQ window number to be configured (0)
 IMAQ zoom factor to be applied to specified window (1)
 Title bar selector (f)
 Hide/show selector (t)
 Embed selector (f)
 Name of panel in which image should be embedded (must be exact)
 Screen coordinates of image (0,0)
 Color Palette
 Error Cluster

Outputs: Error Cluster

3.1.8 **Invert_Revert.VI**



Indigo Systems LabVIEW 7 Toolkit Version 1.10

This VI inverts (flips vertically) or reverts (flips horizontally) an image. The inverted image and reverted image references should be wired to the IMAQ create image VI, and the image in should be wired to the image that is to be inverted or reverted. The invert and revert selectors are independent. Neither, either, or both functions can be selected during operation.

Inputs: Invert image selector (f)
 Revert image selector (f)
 Image in*
 Inverted Image Reference*
 Reverted Image Reference*
 Error Cluster

Outputs: Inverted/reverted Image out
 Error Cluster

3.1.9 Load Palette



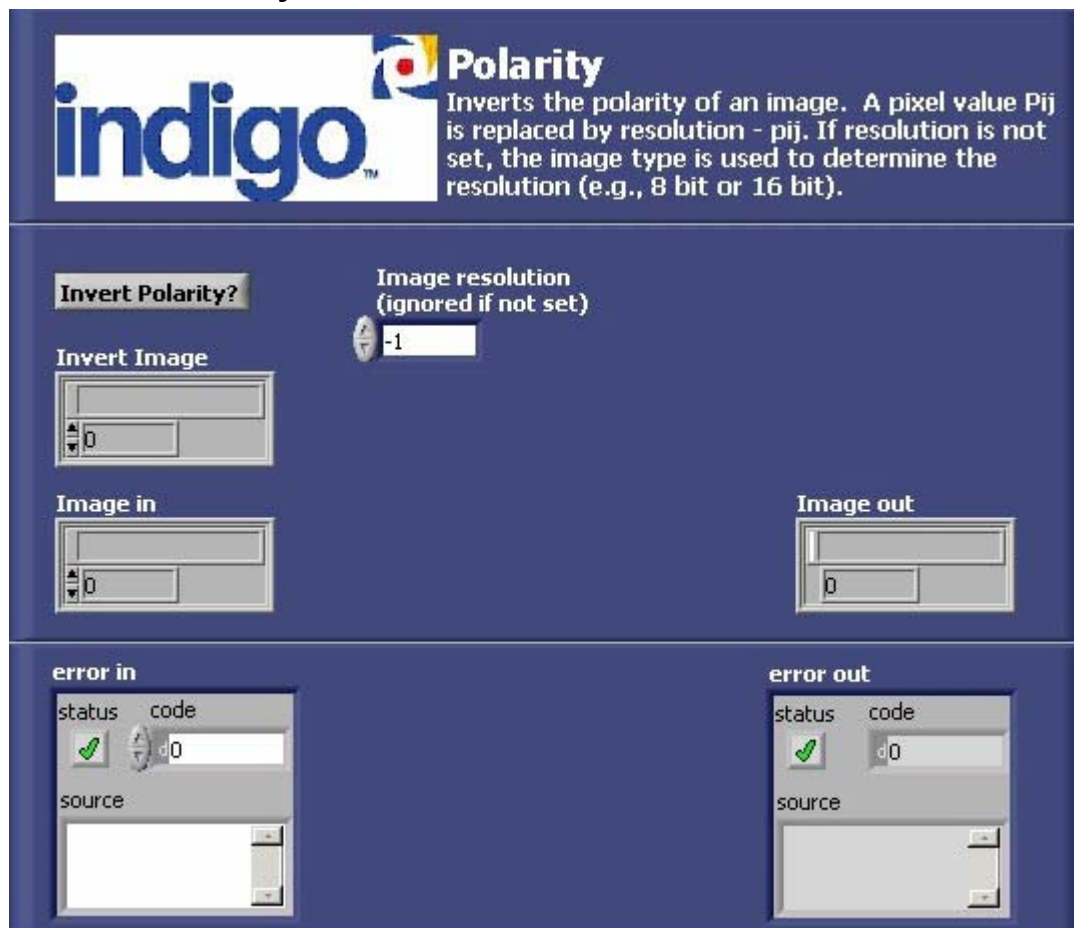
Indigo Systems LabVIEW 7 Toolkit Version 1.10

This VI uses VIs in the Thermoteknix Palette Utilities.llb to load color palettes from Thermoteknix .def files.

Inputs: Path to .def file
Error Cluster In

Outputs: Palette
Inverted Palette
Error Cluster Out

3.1.10 Polarity.VI



This VI inverts the polarity of an image so that high intensity pixels become low intensity pixels and vice-versa. This VI can accept 8 bit, 12 bit, 14 bit, or 16 bit images. With an 8 bit image there is no ambiguity. The data type of the image defines the resolution and there is no need to wire the image

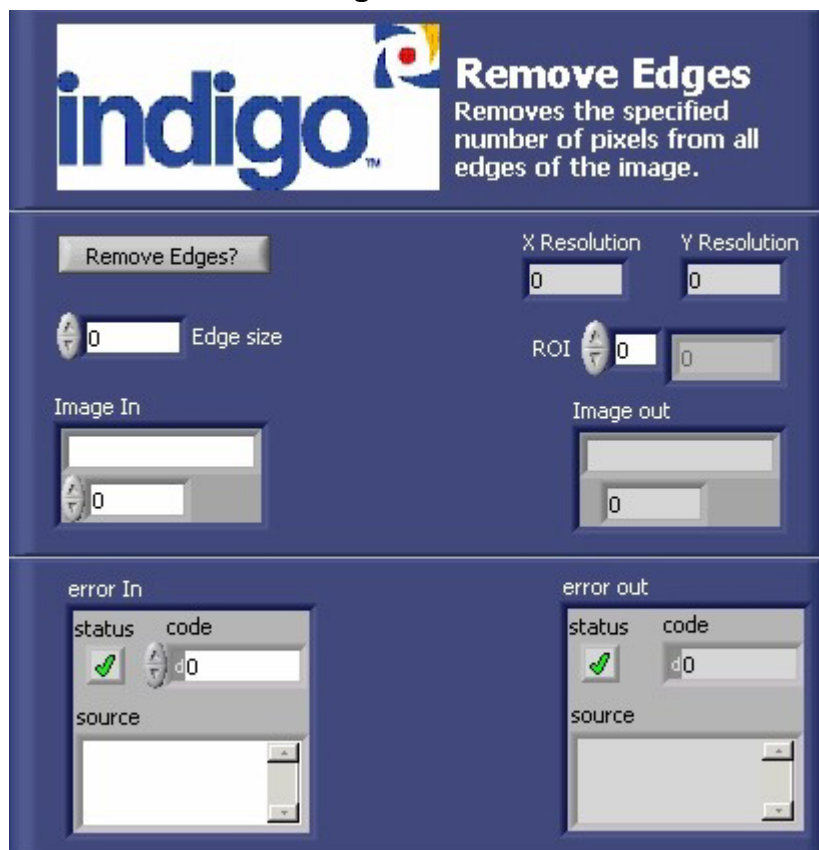
Indigo Systems LabVIEW 7 Toolkit Version 1.10

resolution input. However, 12, 14, and 16 bit images all use the I16 data type and thus it is necessary to wire the image resolution to 4096 (12 bit), 16384 (14 bit), or 65536 (16 bit). If a 12 bit image is fed into this VI but the image resolution is not set then the image resolution will default to 65536. Pixels in the original image that have a value of zero would be assigned the intensity 65535 in the output image, while pixels with the 12 bit maximum intensity of 4095 would be assigned the intensity $65535 - 4095 = 61440$ in the output image. The inverted image input should be wired to the IMAQ create image VI with the appropriate image type selected.

Inputs: Image in*
 Invert image* (Reference to inverted IMAQ image)
 Image resolution (-1, needed only for 12 bit or 14 bit images)
 Error Cluster

Outputs: Inverted polarity image out
 Error Cluster

3.1.11 Remove Edges.VI

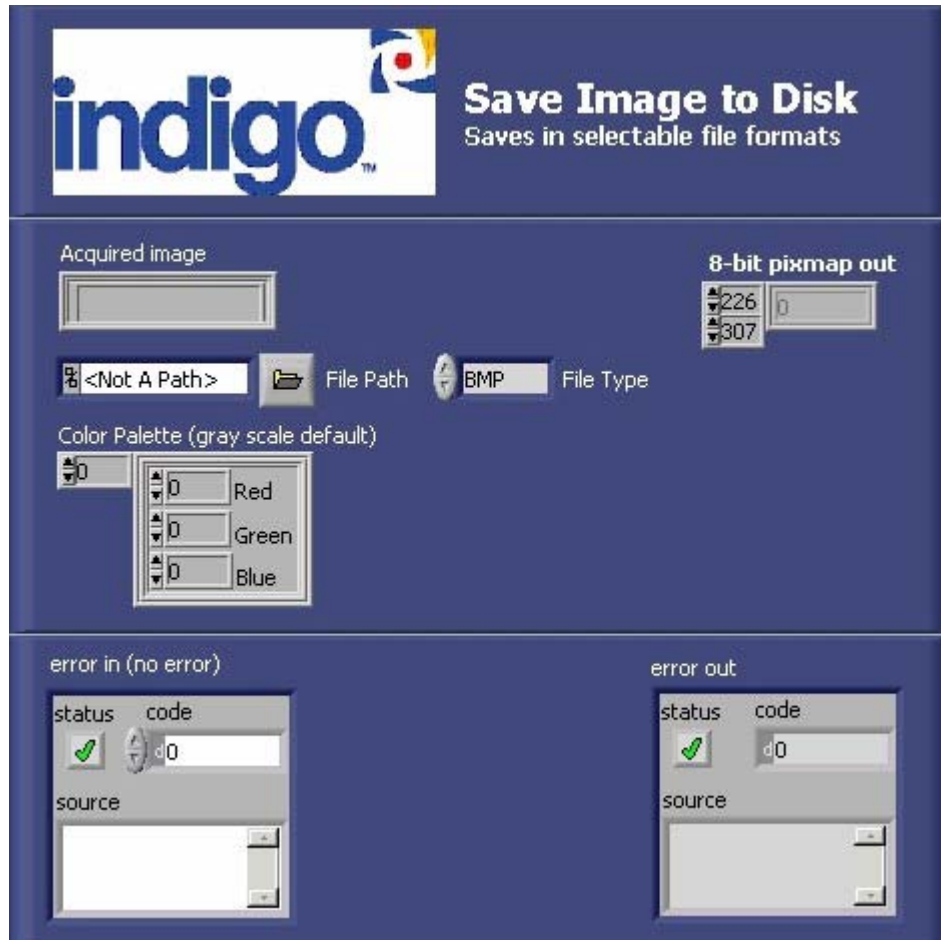


Indigo Systems LabVIEW 7 Toolkit Version 1.10

This VI removes the specified number of pixels from each edge of the input image. This is often employed to remove defective pixels that are frequently found near the edge of a focal plane. Edge removal is included in  **Image Acquisition Setup.VI** which uses this routine. It is also possible to turn on or off edge removal, or change the size of the edge to be removed, while acquisition is in progress. However, some care must be taken when using this VI in combination with  **Image Compression.VI**. If the size of the image is changed during acquisition and image intensity compression or “windowing” is employed, it is essential that  **Image Compression Init.VI** be run before the resized image is subjected to data compression. If this is not done the likely result is a system crash with loss of any unsaved LabVIEW code. In  **Camera GUI.VI** in Toolkit **Example.llb** this problem is solved by applying edge removal after data compression. Data compression always takes place on the full-size image and the edges are removed just before final display. This approach has the advantage of simplifying the code. On the other hand, reinitializing the dll and applying compression to the smaller image has the advantage of increasing the frame acquisition rate.

Inputs:	Image in* Edge removal selector (f) Edge size in pixels (0) Error Cluster
Outputs:	Image with or without edges ROI X resolution Y resolution Error Cluster

3.1.12 Save to Disk



This VI saves an image to disk in the selected format. The standard .bmp, .jpg, .png, and .tiff formats are currently supported. The color palette defaults to standard gray scale (white hot) if the palette connector is unwired. A file dialog will be called if the file path is not wired. Note that 12 bit images will be compressed to 8 bits for saving to disk. This compression is accomplished by dividing all of the 12 bit intensity values by 16. The resulting image pixels are provided as an output array. If the input image is already 8 bit no further compression will be applied.

Inputs: Image to be saved*
 File Path (dialog if not connected)
 File Type (.bmp)
 Color Palette
 Error Cluster

Outputs: Array of 8 bit image pixels
 Error Cluster

3.1.13 Single Image.VI



This VI acquires a single image from the specified IMAQ resource. The image can be displayed to an IMAQ image window and/or saved to disk in .jpg, .png, .bmp, and .tiff formats. Note that 12-bit

images are compressed to 8-bits for saving to disk. See  **save image to disk.VI** for additional information. Edge removal is available as well. If this image is displayed to an IMAQ image window the IMAQ defaults will be used unless this window has been previously formatted using

 **Image Display Setup.VI**. For applications in which acquisition speed is not an issue this VI may be a satisfactory acquisition routine. Tremendous increases in acquisition speed are possible using the lower level setup and acquire routines in the main library. Outputs of **Single Image.VI** include a Boolean variable that indicates whether an image has successfully been acquired, the number of columns and rows in the image (after edge removal if selected), and the image itself. This VI is thus useful for checking basic camera operation in the initialization portion of a more complex

application requiring faster acquisition. See  **Camera GUI.VI** in **Toolkit Examples.llb** for an

**Indigo Systems LabVIEW 7 Toolkit
Version 1.10**

example. Also, the tutorial portion of this manual includes an exercise that uses this VI to build a simple application.

Inputs:	Interface Name* (often img0)
	Image Name (IndigoImage)
	Save image to disk (f)
	File path (dialog if empty)
	Image file type (.bmp)
	Display image (f)
	IMAQ Window for display (0)
	Edge removal (f)
	Edge size, Pixels to remove from edge (0)
	Error Cluster
Outputs:	Acquired image
	Image acquired
	Image rows
	Image Columns
	Error Cluster

3.1.14 Set Color Palette.VI



This VI allows selection of color palettes either from the  Indigo Palettes Global.VI or from Thermoteknix.def files. This VI uses a shift register to pass the current color palette unless the user has requested a change. When a change has been requested the new palette is loaded. This structure

allows  Set Color Palette.VI to be used within an acquisition loop without significant loss of execution speed.

Inputs:

- Indigo palette name
- Path to .def file (dialog if empty)
- Invert palette (f)
- Load Thermoteknix (f)

Outputs:

- Palette name
- Palette
- Error Cluster

3.2 Thermoteknix Palette Utilities.llb

This section of the manual describes the use of Thermoteknix color palettes with Indigo Systems cameras.

Thermoteknix color palettes are contained in text files with the extension .def. These files can be opened with standard text editors and word processors. An example Thermoteknix palette is shown below. Annotations added to the file are set off by square brackets and green text. Additional information about Thermoteknix color palettes can be found at www.thermoteknix.com.

Graded Rainbow

```
8 [Number of fixed points in the palette]
[Intensity, red, green, blue, fill method]
0 0 0 0
36 255 0 255 1
73 0 0 255 1
109 0 255 255 1
146 0 128 0 1
182 255 255 0 1
219 255 0 0 1
255 255 255 255 1
```

The first line contains the name of the palette. The second line contains the number of fixed points in the color palette, and the remainder of the palette definition consists of 8 bit data used to assign RGB values to image pixels with particular 8 bit intensity values. For example, the Graded Rainbow palette assigns to image pixels with intensity value 36 a red value of 255, a green value of 0, and a blue value of 255. Thus an image pixel with intensity 36 would appear purple after application of this palette. The last number on each line determines how RGB values should be assigned to intensities between the fixed points in the palette. If the fill method is “0” then intensities between fixed points are assigned RGB values equal to those at the next higher fixed point. Thus pixels with intensities between 1 and 36 would be assigned RGB values 255, 0, 255 in this palette. If the fill method is “1” then RGB values are assigned to intermediate intensities via linear interpolation.

Software Toolkit VIs for use of Thermoteknix Palettes

The Indigo Systems LabVIEW Software Toolkit includes a set of VIs designed to facilitate use of Thermoteknix color palettes. Each is described below. See also **Example 4.VI** and **Camera GUI.VI** in **Toolkit Examples.llb**.

3.2.1 Load Thermoteknix Palette GUI.VI



This VI includes a simple graphical user interface and features file dialogs. It can be used as a pop-up window from an application, but it can also be used as a stand-alone application. When this VI is run, the user is queried for a path to a .def file. The entire .def file is then loaded, and the names of the Thermoteknix color palettes it contains are extracted and presented to the user in the text ring. The user then selects a color palette and hits load. The equivalent IMAQ palette is calculated and is presented to the user at the right.

Inputs: Path to .def file (dialog if empty)
 Palette Selector
 Load (f)
 Error Cluster

Outputs: Equivalent IMAQ color palette
 Error Cluster

SubVIs

The subVIs found in Load Thermoteknix Palette GUI.VI can be used separately to construct custom routines that load Thermoteknix color palettes. Each of these subVIs is described below.

3.2.2 Load Thermoteknix Palette File.VI

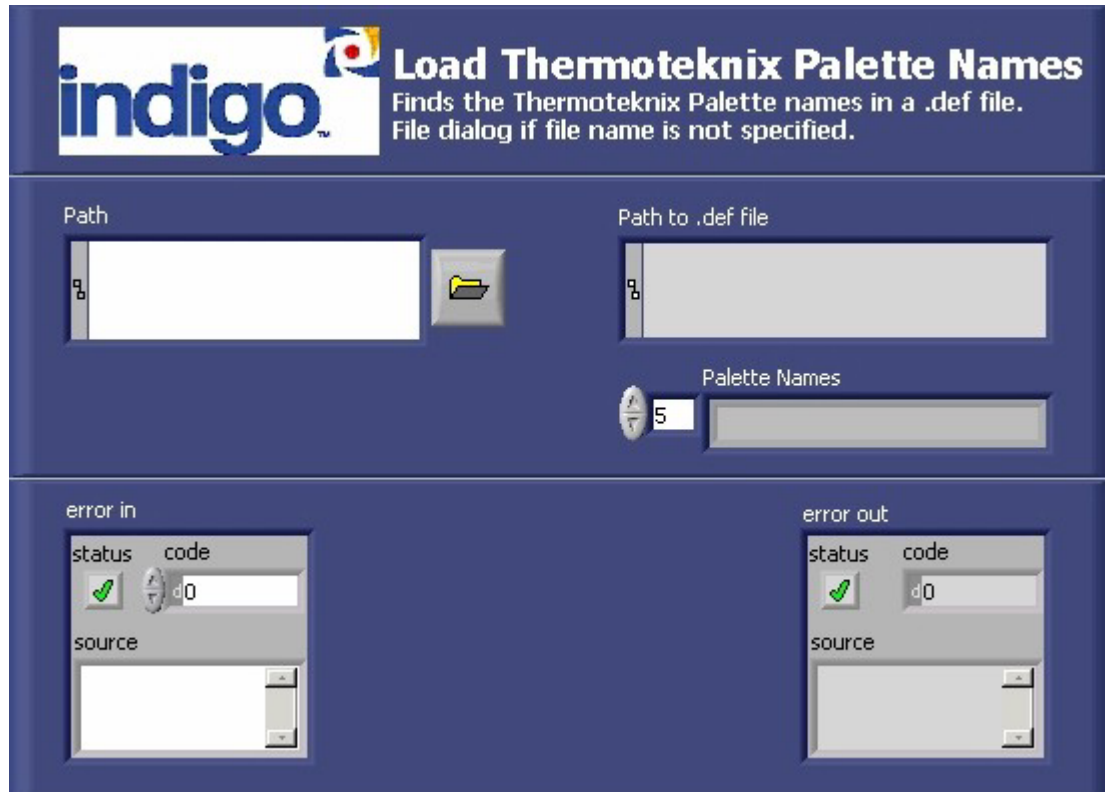


This VI loads an entire .def file (these files are not very large). The text array at right contains the entire text of the file, one line per array element. The file path can be specified as an input to this VI. If the input file path is empty or not a path a file dialog will be called. In either case the path to the loaded .def file is provided as an output.

Inputs: Path to .def file (dialog if empty)
Error Cluster

Outputs: Path to .def file (not a path if cancelled)
Array containing all lines of text from .def file.
Error Cluster

3.2.3 Find Thermoteknix Palette Names.VI

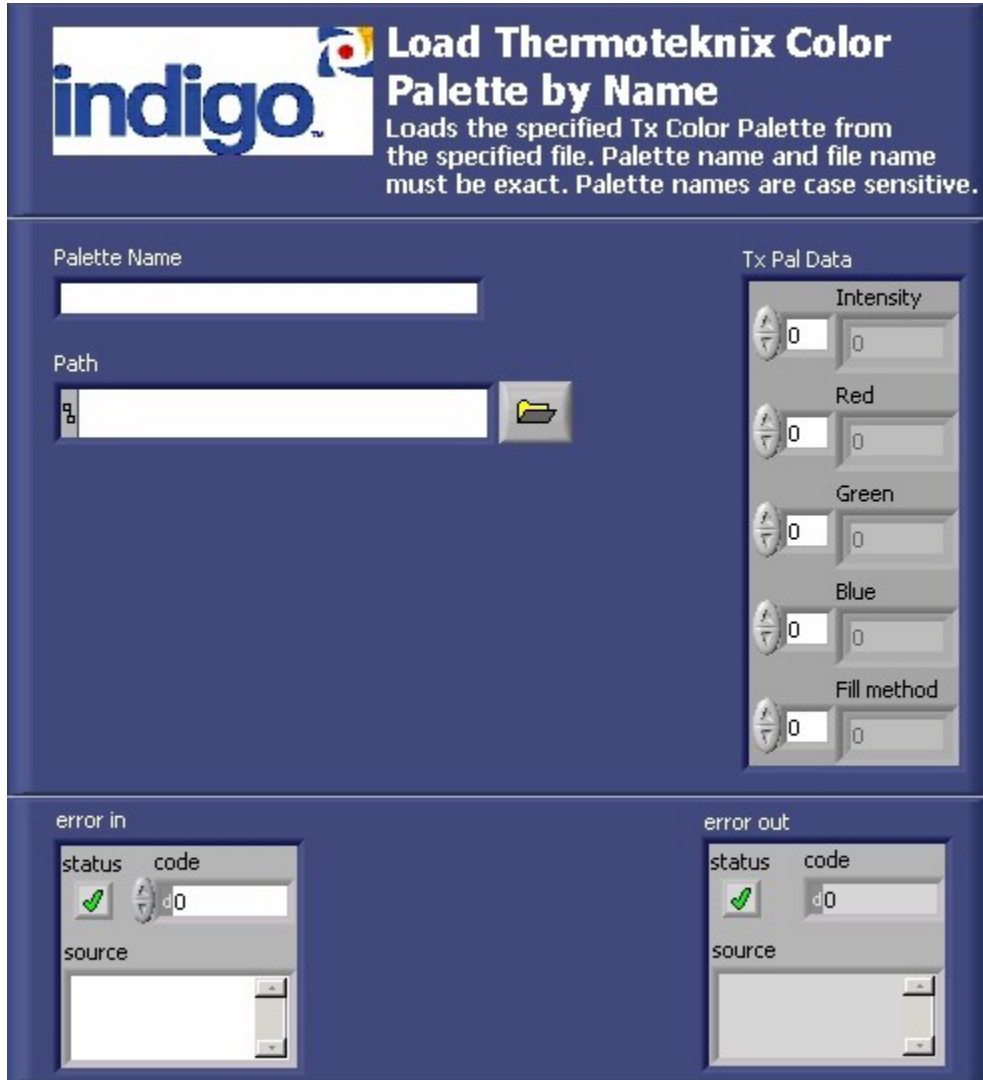


This VI uses  **Load Thermoteknix Palette File.VI** as a subVI to load an entire .def file. The names of the Thermoteknix color palettes contained in the file are then extracted and reported to the user in the string array at right.

Inputs: Path to .def file (dialog if empty)
Error Cluster

Outputs: Path to .def file (not a path if cancelled)
Array of palette names from the .def file.
Error Cluster

3.2.4 Load Thermoteknix Color Palette by Name.VI



Load Thermoteknix Color Palette by Name
Loads the specified Tx Color Palette from the specified file. Palette name and file name must be exact. Palette names are case sensitive.

Palette Name

Path
 

Tx Pal Data

Intensity
 0

Red
 0

Green
 0

Blue
 0

Fill method
 0

error in

status  code 0

source

error out

status  code 0

source

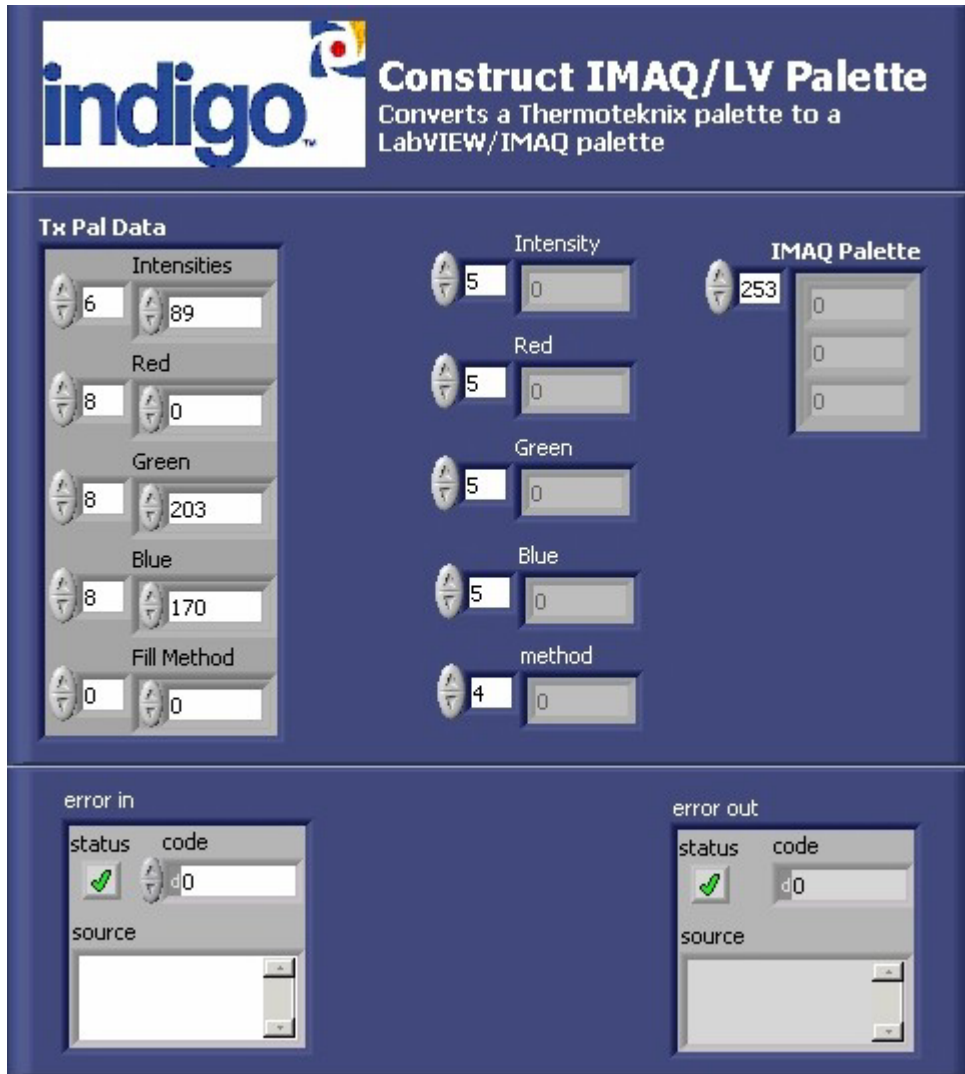
Once  **Find Thermoteknix Palette Names.VI** has been used to find the available palettes in a .def file,  **Load Thermoteknix Color Palette by Name.VI** can be used to load a particular palette. If the names were extracted from the .def file by **Load Thermoteknix Color Palette by Name.VI**, they are automatically exact and loading proceeds with little difficulty. The palette data is reported in the cluster at right in the form of five arrays. The first is an array of the intensities of the fixed points. Then there are arrays with the corresponding RGB values, and finally an array of fill methods. These can be extracted from the cluster using the LabVIEW unbundled tool as usual.

Inputs: Path to .def file (dialog if empty)

**Indigo Systems LabVIEW 7 Toolkit
Version 1.10**

Palette Name (must be exact)
 Error Cluster
 Outputs: Cluster containing Thermoteknix palette data.
 Error Cluster

3.2.5 Construct IMAQ Palette.VI



The front panel of the **Construct IMAQ/LV Palette** VI is divided into three main sections. The top section features the Indigo logo and the title "Construct IMAQ/LV Palette" with a subtitle "Converts a Thermoteknix palette to a LabVIEW/IMAQ palette". The middle section, labeled "Tx Pal Data", contains input controls for "Intensities" (a numeric array with values 6 and 89), "Red" (a numeric array with values 8 and 0), "Green" (a numeric array with values 8 and 203), "Blue" (a numeric array with values 8 and 170), and "Fill Method" (a numeric array with values 0 and 0). To the right of these are five numeric controls for "Intensity", "Red", "Green", "Blue", and "method", each with a value of 0. The bottom section contains two error clusters, "error in" and "error out", each with a status indicator (a green checkmark), a code field (containing "d0"), and a source field.

Once  **Load Thermoteknix Color Palette by Name.VI** has been used to obtain the Thermoteknix palette data it can be passed to  **Construct IMAQ Palette.VI** which will then perform the calculations required to produce an equivalent IMAQ palette. The Thermoteknix palette data is passed to this VI as five arrays containing the intensities, Red, Green, Blue and fill methods

**Indigo Systems LabVIEW 7 Toolkit
Version 1.10**

for the fixed points in the palette. These arrays should be bundled into a cluster in the order shown on the panel.

Inputs: Thermoteknix Palette Data
Error Cluster
Outputs: IMAQ color palette
Error Cluster

3.2.6 **Make Color Table.VI**



This VI produces a color table for use with LabVIEW pictures from an IMAQ color palette. This is particularly useful when LabVIEW picture controls are used on a front panel in place of IMAQ image windows, and is needed when using LabVIEW image file saving routines instead of the IMAQ image file saving routines.

Inputs: IMAQ Color Palette
Error Cluster
Outputs: LabVIEW Color Table
Error Cluster

3.3 Non-Uniformity Correction and Bad Pixel Replacement

This section of the manual describes the use of Non-Uniformity Correction (NUC) and Bad Pixel Replacement (BPR) routines contained in the Toolkit. The NUC and BPR algorithms are contained in dll's called by the VIs provided in the LabVIEW library **NUC and BPR.lib**. The library contains

three high-level VIs:  **Initialize NUC and BPR.VI**,  **NUC and BPR.VI**, and  **Close NUC_BPR Library.VI**. The library also contains a larger number of low-level VIs. Most programmers will use only the high-level VIs. A description of the algorithms follows.

Non-Uniformity Correction (NUC)

Pixels in a focal plane typically have a range of gains and offsets. Indigo Systems IRVista software includes a utility that allows the user to measure the gain and offset of each pixel in the focal plane and then construct a settings file that contains a gain and offset correction for each pixel. Many Indigo cameras are supplied with one or more settings files. Corrected pixel values are then calculated according to

$$P_{ij}^{corrected} = G_{ij} * P_{ij} + O_{ij}$$

Note that G_{ij} and O_{ij} are not the measured gains and offsets of the pixels but rather correction terms to make the gains and offsets uniform across the focal plane.

Bad Pixel Replacement (BPR)

Most focal planes contain a few defective pixels that cannot be corrected by the NUC routines. The data from these pixels are typically replaced by data from one or more nearby pixels. The algorithm employed in the Toolkit averages the nearest good pixel in each of four directions (+x, -x, +y, -y) to replace the data from the bad pixel. For example, in the partial image below the pixels marked with X's are defective while the rest are good.

1	2	3	4	5	6	7	8	9
2								
3			X	X				
4								
5								
6								
7								

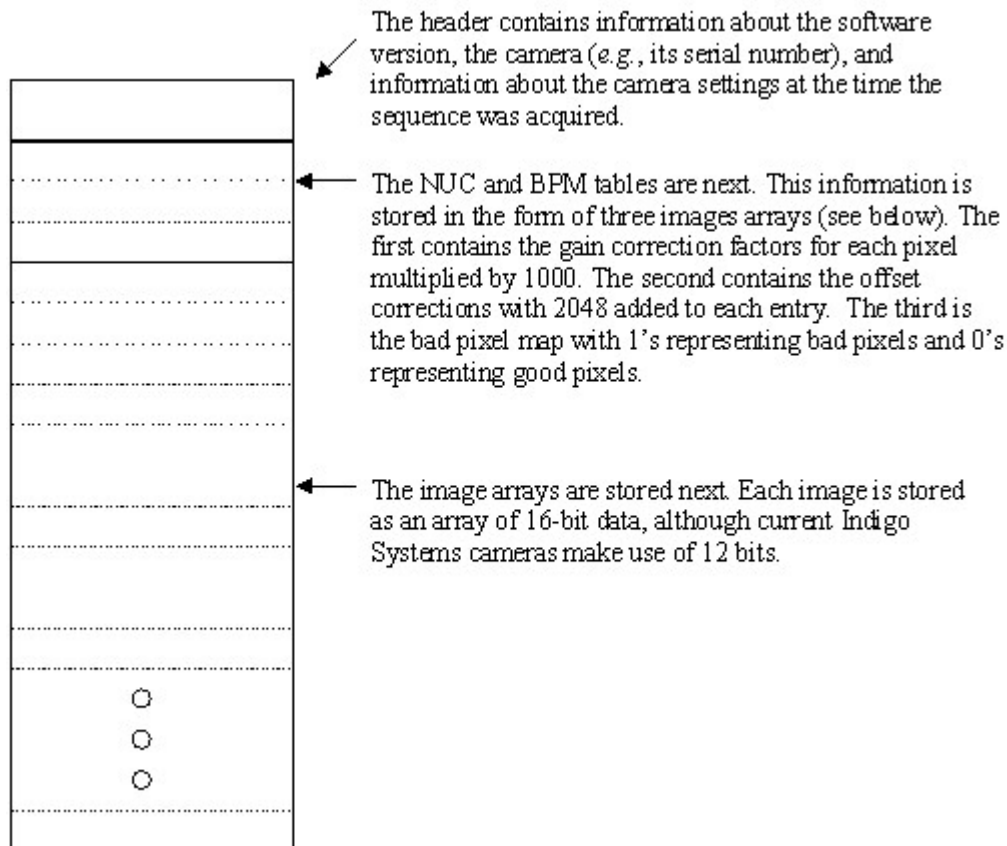
In this example pixels (3,4) and (3,5) are bad. Pixel (3,4) would be replaced by the average of pixel (3,3); (3,6); (2,4); (4, 4). Similarly, pixel (3,5) would be replaced by the average of pixels (3,3); (2,5); (4,5); (3,6).

Indigo Systems LabVIEW 7 Toolkit Version 1.10

NUC and BPR Data: The Indigo Settings Files

NUC and BPR data is specific to a particular focal plane. During manufacture by Indigo Systems, each focal plane is carefully calibrated against uniform sources of known intensity. The results of the calibration include a gain and offset for each good pixel in the focal plane, as well as a map of the bad pixels. These are provided in the form of a settings file. In turnkey Indigo Systems software packages such as IRVista (see www.indigosystems.com), an image sequence file is simply a settings file with images appended. The structure of Indigo settings and sequence files is as follows.

An image sequence or settings file is a LabVIEW standard I16 data file. It consists of three sections: the header, the NUC and BPR tables, and the images. A settings file is simply an image sequence file with zero viewable images, and thus we do not distinguish between images sequences and settings file in the discussion that follows. Each of the three sections is described briefly in the figure and then in more detail.



The header portion of the sequence file contains the software and hardware settings in use at the time the image sequence was acquired.

Header elements and comments (each element is a 16 bit or 2 byte integer):

0 - major software version number

Indigo Systems LabVIEW 7 Toolkit
Version 1.10

- 1 - minor software version number
- 2 - number of elements in header
 - Allows for future expansion of the header file while maintaining compatibility with older sequence and settings files.
- 3 – columns in each image
- 4 – rows in each image
- 5 - images lsb or least significant byte
- 6 - images msb or most significant byte
 - The number of images is header [6] * 2¹⁶ + header [5]
- 7 - NUC/BPR elements lsb
- 8 - NUC/BPR elements msb
 - The number of elements in the NUC/BPR tables is header [8] * 2¹⁶ + header [7]
- 9 - 12 Camera PN
- 13 - 16 Camera SN
- 17 - 20 Camera VN
- 21 - 24 Camera options
- 25 - 26 Case Temp
- 27 - 28 FPA Temp
- 29 - 30 FPA Mode
- 31 - Long Int Mode
- 32 - Ext Int time
- 33 - 34 Int Time
- 35 - Short Int Time
- 36 - 37 VDetCom
- 38 - 61 not in use
- 62 - Integration time us (lsb)
- 63 - Integration time us (msb)
 - The integration time in microseconds is header [63] * 2¹⁶ + header [62]
- 64 – Sync Mode (0 Internal, 1 External).

The next section contains the NUC and BPR tables. The data is stored as a set of three image arrays. The first one is the gains table, then the offsets, then the bad pixel map. The tables are identical in size and format to viewable images. If the image has M columns and N rows, it will have M x N elements, each two bytes. For example, if the image is 316 x 252 pixels then each image occupies 79632 elements or 159264 bytes. Thus the first 130 bytes of the settings file are the header, the next 159264 bytes are gains, then 159264 bytes of offsets, then 159264 bytes of bad pixels map.

The gains are all multiplied by 1000. In other words, an entry in the gain table of 1050 represents a gain factor of 1.05. Similarly, 2048 is added to each offset. In other words, an entry of 2059 in the offset table represents an offset of 11. This allows the gains and offsets to be stored as 16-bit unsigned integers. In the bad pixel map, bad pixels are ones and good pixels are zeros.

Indigo Systems LabVIEW 7 Toolkit
Version 1.10

During operation the image data is corrected as follows:

$P_{ij,corrected} = G_{ij} * P_{ij} + O_{ij}$ where P_{ij} is the column i , row j pixel, G_{ij} refers to gains and O_{ij} refers to offsets.

The viewable images are stored sequentially following the NUC and BPR tables, each occupying $M \times N \times 2$ bytes.

Details of Data Format

Actual 2D image data, whether viewable or part of the NUC/BPR portion of the file, is written as a long string of bytes using the standard LabVIEW Write I16 file command. The data is written as $r0\ c0; r0\ c1; \dots r0\ cM; r1\ c0; r1\ c1; \dots r1\ cM; \dots rN\ cM$ where $r0$ refers to row 0 and $c1$ refers to column 1. If the data is read back using the standard LabVIEW read characters function that data will appear in this order in hex format.

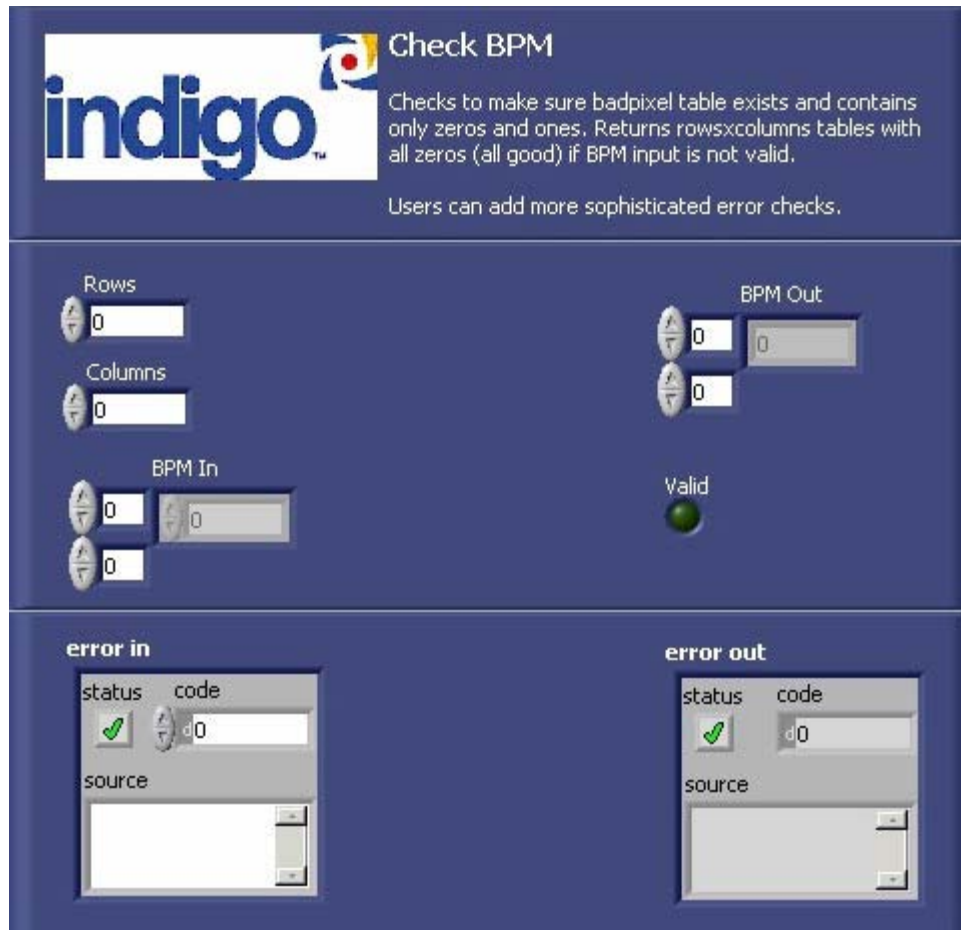
Note that a common source of confusion with LabVIEW data files is that data is stored “backwards” as “bigendian” numbers. This is because LabVIEW was originally written for Mac OS.

LabVIEW: "BIG ENDIAN"	(00000001	(00000000	(Byte4:Byte3:Byte2:Byte1
	)	00000001)	)
(Win) C: "LITTLE ENDIAN"	(00000001	(00000001	(Byte1:Byte2:Byte3:Byte4
	)	00000000)	)

The remainder of section 3.3 describes the individual VIs contained in the NUC and BPR library.

Indigo Systems LabVIEW 7 Toolkit Version 1.10

3.3.1 Check BPM.VI



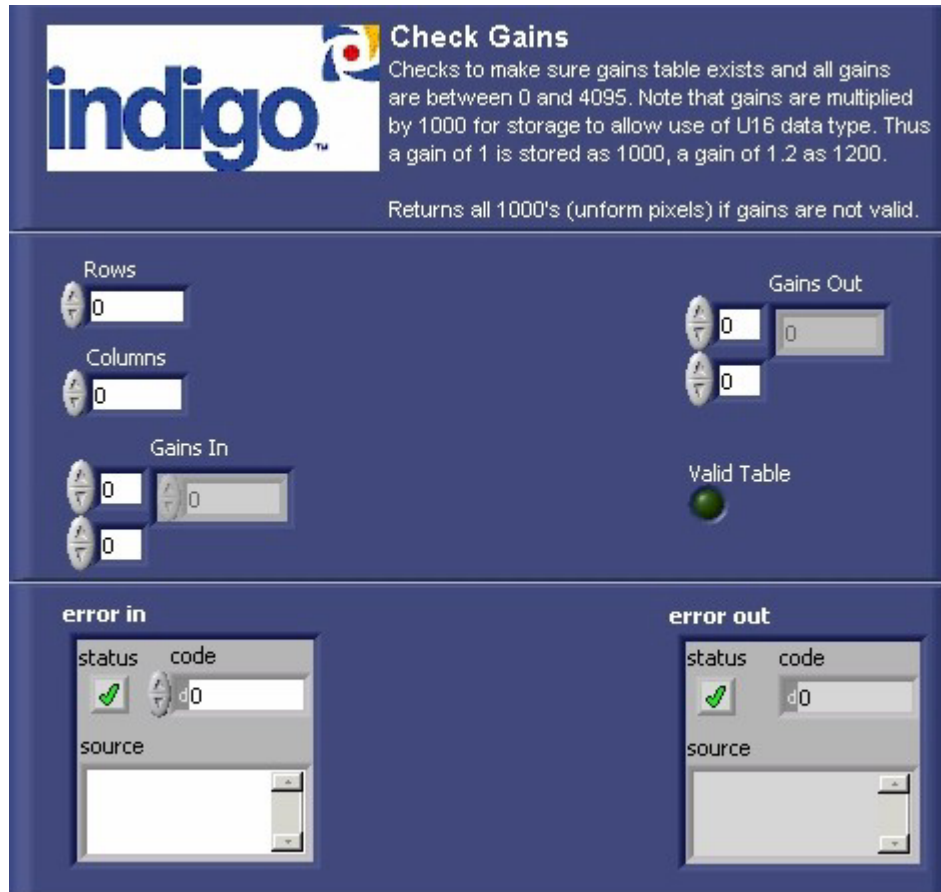
This VI performs basic error checks on the bad pixel map data. For example, all of the elements in the table should be either zeros or ones. If the submitted bad pixel map is not valid, a blank bad pixel map with all zeros (all good pixels) of size rows x columns is returned.

Inputs: Rows*
Columns*
Bad Pixel Map*
Error Cluster

Outputs: Bad Pixel Map
Valid (true if submitted table is valid)
Error Cluster

Indigo Systems LabVIEW 7 Toolkit Version 1.10

3.3.2 Check Gains.VI



This VI performs basic error checks on the gains table. For example, all gains should be between 0 and 4095. If the submitted gains table is not valid, a blank gains table with all elements equal to 1000 (or a gain correction of one) of size rows x columns is returned.

Inputs: Rows*
 Columns*
 Gains Table*
 Error Cluster

Outputs: Gains Table
 Valid (true if submitted table is valid)
 Error Cluster

Indigo Systems LabVIEW 7 Toolkit Version 1.10

3.3.3 Check Offsets.VI

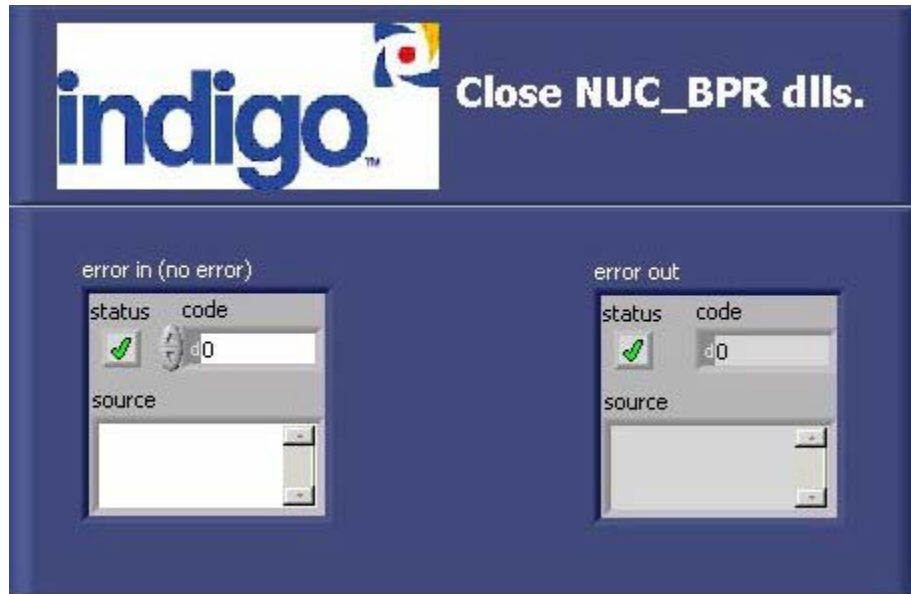


This VI performs basic error checks on the offsets table. For example, all offsets should be between 0 and 4095. If the submitted offsets table is not valid, a blank offsets table with all elements equal to 2048 (or an offset correction of zero) of size rows x columns is returned.

Inputs: Rows*
 Columns*
 Offsets Table*
 Error Cluster

Outputs: Offsets Table
 Valid (true if submitted table is valid)
 Error Cluster

3.3.4 Close NUC_BPR Library.VI



This VI closes the dll used for NUC and BPR. This VI should be run once as the application closes

if  **Initialize NUC and BPR.VI** and  **NUC and BPR.VI** are used. See the discussion of  **Camera GUI.VI** and  **Example 5.VI** in Chapter 2 for examples of the use of NUC and BPR VIs.

Inputs: Error Cluster
Outputs: Error Cluster

3.3.5 Initialize NUC and BPR.VI



This VI initializes the NUC and BPR dll. It must be run prior to running  **NUC and BPR.VI**. The initialization of the NUC and BPR dll takes time and should not be done per image. However, initialization of the dll is required anytime the image changes size. When  **Initialize NUC and BPR.VI** is run, the settings file is loaded from the specified path (dialog of path not specified), the tables are checked using  **Check BPM.VI**,  **Check Gains.VI**, and  **Check Offsets.VI**. If the tables are not found an error is returned and the dll is not initialized. If tables are found and are

Indigo Systems LabVIEW 7 Toolkit Version 1.10

valid they are used to initialize the dll. If tables are found but are not valid then blank tables are used to initialize the dll.

Inputs: Path to camera settings file (Dialog if not specified)
Error Cluster

Outputs: Bad Pixel Map
Gains Table
Offsets Table
Rows in image
Columns in image
Valid (true if submitted tables are valid)
Error Cluster

3.3.6 NUC and BPR.VI



This VI calls the Bad Pixel Replacement (BPR) and Non-Uniformity Correction (NUC) routines.

To use an alternate NUC/BPR routine, write a new dll that implements the BRP and NUC routines. The input to the call function should be a 12 or 14 bit image and the output should be a corrected 12 or 14 bit image and a corrected image that has been scaled to 8 bit.

Must use init function before using this function!

Caution: An error in the BPR/NUC routines can crash LabVIEW and the computer!

NUC ☒ ON BPR ☒ ON

image pixels in

16b corrected image pixels out

error in

status	code
✓	0

source

error out

status	code
✓	0

source

Indigo Systems LabVIEW 7 Toolkit Version 1.10

This VI applies the Non-Uniformity and Bad Pixel Replacement algorithms to the input image array. The NUC and BPR algorithms are described in the introductory remarks in section 3.3. Note that



Initialize NUC and BPR.VI must be run before this VI is called. Note that passing of gains and offsets tables as well as the bad pixel map from the initialization function to the NUC and BPR functions is internal to the dll and does not need to be wired by the LabVIEW programmer.

Inputs: Input image pixels*
 NUC on
 BPR on
 Error Cluster

Outputs: Corrected image pixels
 Error Cluster

3.3.7 Read BPR from Settings File.VI



Read Bad Pixels Array

BPR File: 

Rows: Columns:

Header:

Bad Pixel Map:

error in (not an error)

status	code
	0

source:

error out

status	code
	0

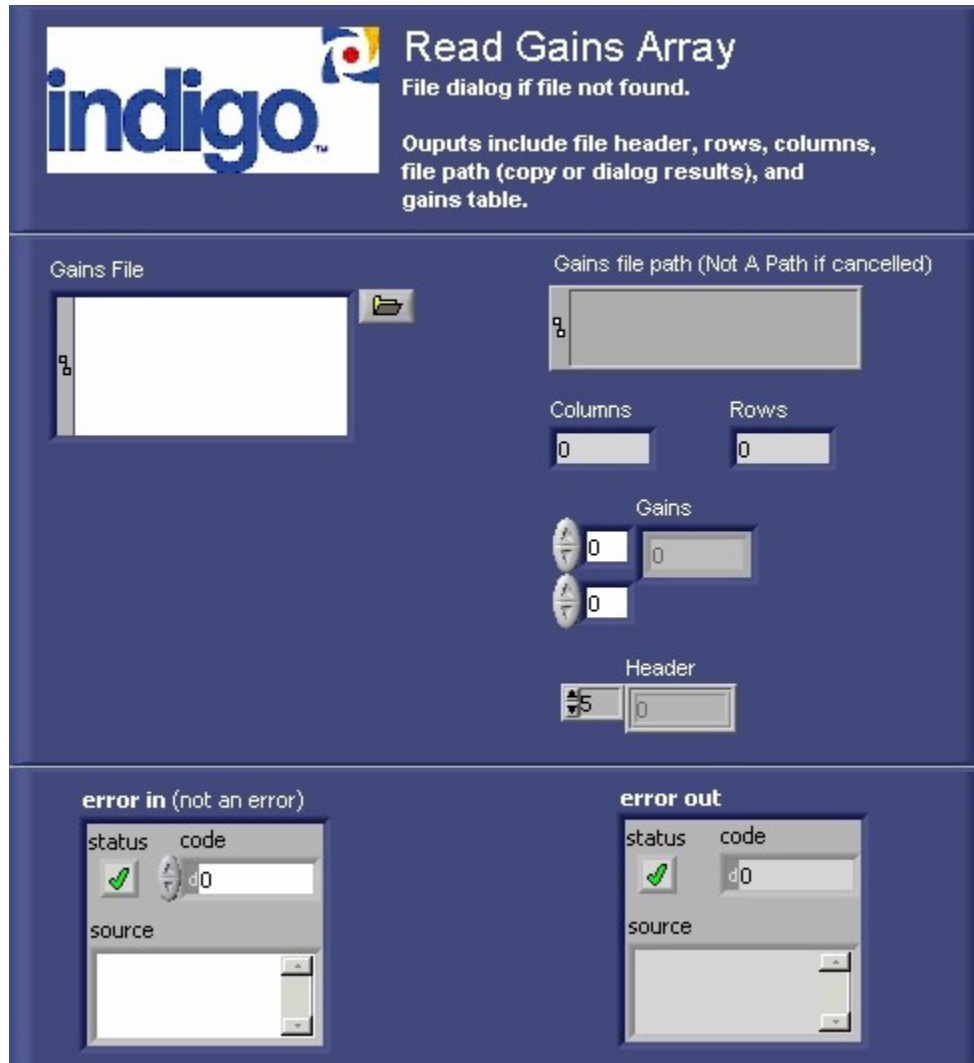
source:

Indigo Systems LabVIEW 7 Toolkit
Version 1.10

This VI loads the Bad Pixel Map from the specified settings file. It is used as a sub-VI by  **Initialize Library.VI** but can also be used as a stand-alone in principle. The file header for the settings file must be read first using  **Read Header.VI** and passed to this VI. Most programmers will not use this VI directly.

Inputs:	BPR File Path File Header Error Cluster
Outputs:	Bad Pixel Map Rows Columns Error Cluster

3.3.8 Read Gains from Settings File.VI



This VI loads the Gains Table from the specified settings file. It is used as a sub-VI by  **Initialize Library.VI** but can also be used as a stand-alone in principle. The file header for the settings file must be read first using  **Read Header.VI** and passed to this VI. Most programmers will not use this VI directly.

Inputs: Gains Table File Path
 File Header
 Error Cluster
Outputs: Gains Table
 Rows

Indigo Systems LabVIEW 7 Toolkit
Version 1.10

Columns
Error Cluster

3.3.9 Read Offsets from Settings File.VI



This VI loads the Offsets Table from the specified settings file. It is used as a sub-VI by  **Initialize Library.VI** but can also be used as a stand-alone in principle. The file header for the settings file must be read first using  **Read Header.VI** and passed to this VI. Most programmers will not use this VI directly.

Inputs: Offsets File Path
File Header
Error Cluster
Outputs: Offsets Table
Rows

Indigo Systems LabVIEW 7 Toolkit Version 1.10

Columns
Error Cluster

3.3.10 Read Header Array Elements from File.VI



This VI loads the Header Array Elements from the specified settings file. It is used as a sub-VI by

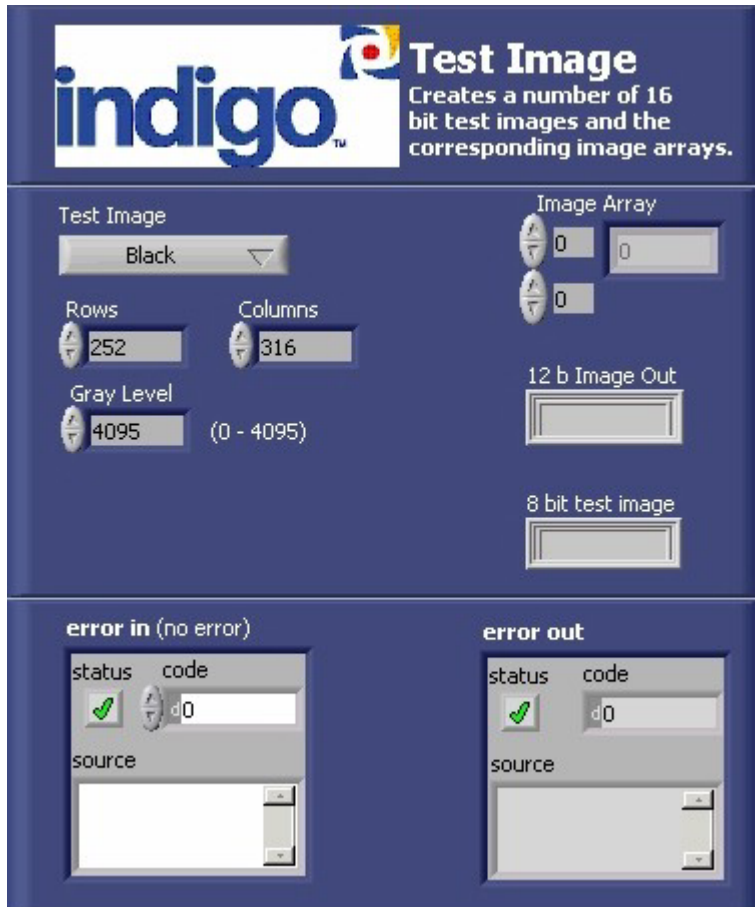
 **Initialize Library.VI** but can also be used as a stand-alone in principle. Most programmers will not use this VI directly.

Inputs: Settings File Path
 Prompt if file not found
 Error Cluster

Outputs: File Path to Settings File
 Elements in Header
 Header
 Error Cluster

3.4 Low-Level Toolkit Utilities

3.4.1 Test Image.VI



This VI produces a variety of simple images and equivalent 16 bit image arrays. It is useful for making reference images and test images, and for debugging some image processing routines. The user can select from among the following test images: Black and gray images are uniform, while the vertical stripes image contains stripes of width four pixels that alternate between the gray level and 0 (black). Horizontal stripes works similarly and cross hatch contains both horizontal and vertical stripes. Note that connecting the 12 bit image directly to IMAQ WindDraw.VI can yield unexpected results because IMAQ WindDraw.VI scales floating point and images with integer pixels greater than 8 bits for display according to

$\text{Display}(x,y) = (\text{src}(x,y) - \text{min}) * 255 / (\text{max} - \text{min})$. Thus a 16 bit image with a uniform gray level of 3000 will appear black when displayed by IMAQ WindDraw.VI. See IMAQ documentation for more details.

**Indigo Systems LabVIEW 7 Toolkit
Version 1.10**

Inputs: Image Rows (252)
 Image Columns (316)
 Gray level (4095)
 Test Image: black, gray, vertical stripes, horizontal stripes, cross hatch. (black)
 Error Cluster

Outputs: Image Array
 8 bit IMAQ test image out
 12 bit test image out
 Image array
 Error Cluster

3.4.2 Invert Palette.VI

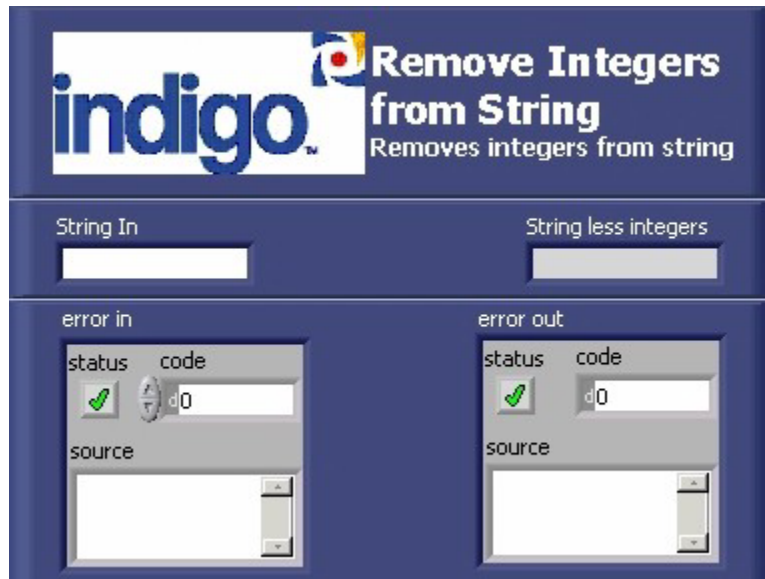


This VI inverts an IMAQ color palette. For example, running this routine on a palette in which red is hot and blue is cold will produce a new palette in which red is cold and blue is hot.

Inputs: IMAQ Color Palette In
 Error Cluster

Outputs: Inverted IMAQ Color Palette Out
 Error Cluster

3.4.3 Remove Integers from String.VI



This VI removes all integers from an input string.

Inputs: String In
Error Cluster

Outputs: String Sans Integers
Error Cluster

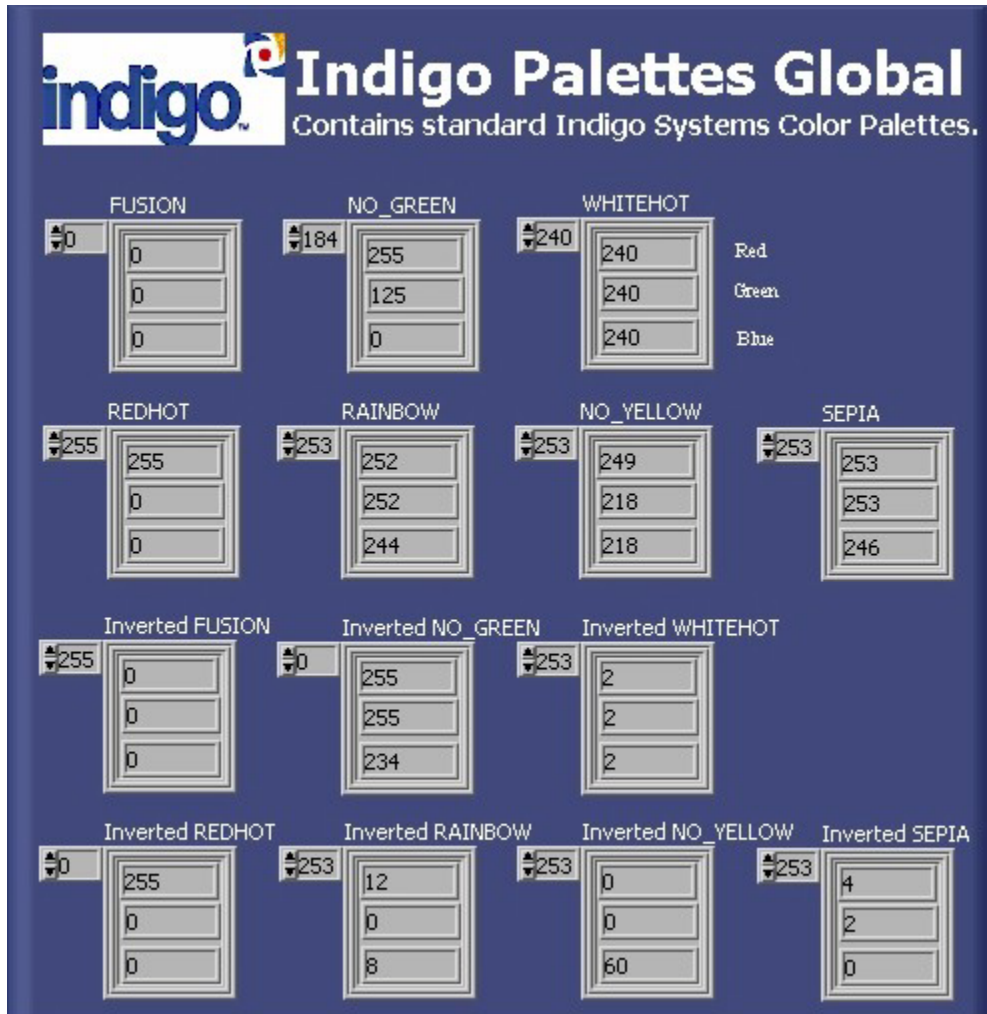
3.4.3.1 IndigoGlobal.VI



This global variable contains references to the current IMAQ image and IMAQ session for use in a variety of VIs.

Indigo Systems LabVIEW 7 Toolkit Version 1.10

3.4.4 Indigo Palettes Global.VI



This global variable contains the standard Indigo Systems color palettes and their inverses. These can be used to apply false color to 8-bit IMAQ images and can be converted to LabVIEW color tables using  **make color table.VI** in the Thermoteknix palette library.

3.5 Other Libraries and VIs in the ToolKit

Extlibsp.lib contains VIs used to embed an IMAQ image in a virtual instrument panel. This library is available at the National Instruments web site, but is included here for convenience. In addition, the VI **Make Child.VI**, available at the National Instruments web site, is included in the **Low Level Toolkit Utilities library**.

4 Error Codes

Error	VI
5001 Image size is zero or no image acquired.	Single Image.VI
5003 Invalid resolution setting	Polarity.VI
5004 Invalid image type for polarity inversion	Polarity.VI
5005 Palette not found.	Load Thermoteknix Palette by name.VI
5006 Tx Palette load cancelled or palette file not found	Load Thermoteknix Palette file.VI
5007 Unsupported image type for save	Save Image to Disk.VI
5009 Invalid Header file. Number of images in header is different from images in file	Read Header Array Elements from File.VI
5010 Invalid header file. Image size is zero.	Read Header Array Elements from File.VI

5 Concluding Remarks

The Indigo Systems LabVIEW Software Toolkit provides a set of functions that are useful when developing applications that use Indigo Systems cameras. Future updates will expand the functionality of the Toolkit. Technical support and custom software development are available from Indigo Systems Corporation at (805) 964-9797, www.indigosystems.com.

***Indigo Systems
LabVIEW Toolkit
For Indigo Alpha NIR Cameras v. 1.1***

6 Introduction

Indigo Alpha NIR LabVIEW Toolkit is a set of image acquisition and camera control subroutines for the LabVIEW/IMAQ programming environment. It is assumed that the user is at least somewhat proficient with the LabVIEW/IMAQ programming environment. Additional information concerning LabVIEW and IMAQ training are available at the National Instruments web page at www.ni.com. Custom software development for selected Indigo Systems imaging products is available from WinSoft Corporation at www.winsoft.com. The block diagrams of many of the low level VIs contained in this toolkit are password protected.

Indigo Alpha NIR LabVIEW Toolkit is most powerful when used in conjunction with the more general *Indigo LabVIEW Toolkit*. The more general toolkit includes many image analysis and processing functions.

6.1 Hardware and Software requirements

To install *Indigo Alpha NIR LabVIEW Toolkit* you will need the following:

- 6.1.1 IBM compatible, Pentium based computer. Pentium III 800 MHz or better recommended.
- 6.1.2 XGA (1024 x 768) or better screen resolution.
- 6.1.3 Windows NT or Windows 2000 operating system.
- 6.1.4 100 MB of available hard drive space. 200 MB or more recommended.
- 6.1.5 256 MB RAM memory. 512 MB recommended.
- 6.1.6 National Instruments LabVIEW 7.0 Express or higher full development version.
- 6.1.7 National Instruments IMAQ version 2.6 or higher and IMAQ Vision Development Module version 7.0 or higher.
- 6.1.8 *National Instruments PCI-1422 Frame Grabber Card.
- 6.1.9 *Indigo Systems Alpha NIR camera.
- 6.1.10 *National Instruments cable for the Alpha NIR camera.

**not required for Indigo Alpha NIR LabVIEW Toolkit installation, but needed for full operation of the software.*

6.2 Installing the Indigo Alpha NIR LabVIEW Toolkit

Indigo Alpha NIR LabVIEW Toolkit is provided on a CD.

**Indigo Systems LabVIEW 7 Toolkit
Version 1.10**

To install, simply copy the Indigo Alpha NIR LabVIEW Toolkit folder to a convenient location on your hard drive. **Right-click on each of the files and deselect “read only.”** Be sure National Instruments LabVIEW, IMAQ, and IMAQ Vision have been installed on the target machine. The toolkit should be installed on the same logical drive as IMAQ and IMAQ Vision.

6.3 Indigo Alpha NIR LabVIEW Toolkit Overview

Indigo Alpha NIR LabVIEW Toolkit is a set of image acquisition camera control subroutines for the LabVIEW/IMAQ programming environment. It is assumed that the user is at least somewhat proficient with the LabVIEW/IMAQ programming environment.

Chapter 2 contains a description of the more commonly used high-level VIs included in the toolkit.

Chapter 3 contains a description of other VIs in the Toolkit.

Chapter 4 lists error codes specific to the Indigo Alpha NIR LabVIEW Toolkit.

Chapter 5 contains concluding remarks and references for further help.

7 Getting Started

7.1 Indigo Alpha NIR LabVIEW Toolkit Overview

The Indigo Alpha NIR LabVIEW Toolkit is contained in the VI library **Alpha NIR Control Kit.llb**. Most programmers will use only two of the VIs in the toolkit: Camera GUI.VI and Camera Control.VI. These are both top-level VIs that will open when the programmer double-clicks on the .llb file. The remaining VIs are lower level subVIs that most programmers will not use. Brief descriptions of the lower level VIs are included in Chapter 3 of this manual.

7.2 Verifying Camera Operation

Before we attempt to use the toolkit we will check to make sure your Indigo Systems Alpha NIR camera and National Instruments Framer Grabber Board are operating properly. To do this, navigate to *NI Measurement and Automation Explorer (MAX)*. There is probably a shortcut to MAX on your desktop. Go to *Devices and Interfaces*, and find your IMAQ board. You should see something like the following:

+ IMAQ PCI-1422: img0.

We are after the interface name that follows the colon. If you used the default settings during installation the interface name is probably img0.

Click on the + and you should see something like the following:

Channel 0: Indigo Systems Alpha NIR.

Click on this and then hit the grab button beneath the menu bar. You should see an image from the camera. If no image is acquired then the camera, board, or related software have probably not been properly installed. Please reinstall and try again.

7.3 Top Level VIs

Double click on the library **Alpha NIR Control Kit.llb**. The VIs  **Alpha NIR GUI.VI** and

Alpha  NIR Control.VI will launch. The **Alpha NIR GUI.VI** control panel pops up when the VI runs, presenting the user with a simple GUI for controlling the most common camera functions. It is designed to be used as a stand-alone application or can be called from a larger application. **Alpha**

Indigo Systems LabVIEW 7 Toolkit Version 1.10

NIR Control.VI is designed to be used as a subroutine to control the camera when user input is provided separately or is not required. Camera control and settings parameters are passed via the connector pane to **Alpha NIR Control.VI**.

7.3.1 Alpha NIR GUI.VI



Running Alpha NIR GUI.VI

Run the VI. A pop-up window will ask you to enter the name of the interface you want to acquire data from. The name of the interface was determined in section 2.2 above and is probably img0. Type in the name continue. After a few second delay the current camera settings are displayed on the front panel along with a continuously acquired image from the camera. If the camera was in external sync mode it was reset to internal sync. Change some of the settings such as image orientation, video mode, and so on. Observe the results. Hit the FPA and Status buttons and observe the pop-up panels that display additional camera parameters such as the contents of the FPA register. Additional information about the FPA register can be obtained from the Alpha NIR icd file. This VI can be used as a subVI. When it is called the front panel will open, allowing the user to control the camera.

Indigo Systems LabVIEW 7 Toolkit Version 1.10

IMAQ Sessions

In order to communicate with a camera, an IMAQ session must be opened. There are two ways to do this using  **Alpha NIR GUI.VI**. Although the controls and indicators are not shown on the front panel, this VI allows the programmer to wire an IMAQ Session In to this VI when it is used as a subVI, and the IMAQ session out can then be passed to the next VI. If the IMAQ Session In terminal is not wired the user will be queried for an interface name and an IMAQ session will be opened. This new IMAQ session will be closed when the user hits the exit button if Pass Session is wired to false or is not wired. If Pass Session is set to true the IMAQ session will be passed out of this VI. It can be used by other IMAQ functions, but it should be closed before the calling application exits.

Reserved IMAQ Image

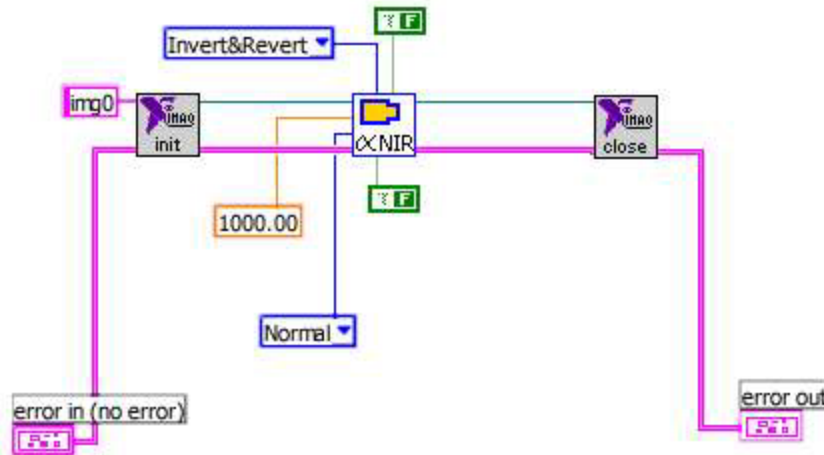
This VI uses IMAQ Image number 15. It is recommended that programmers do not use IMAQ Image number 15 elsewhere in their code if they are using  **Alpha NIR GUI.VI**.

Inputs:	IMAQ Session In
	Pass Session
	Error In
Outputs:	IMAQ Session Out
	Error Out

7.3.2 **Alpha NIR Control.VI**



Minimize or close **Alpha NIR GUI.VI** and open **Alpha NIR Control.VI**. This VI will not run unless the IMAQ Session In is wired. To use the VI, place it on a block diagram as follows:



In this example we have initialized the IMAQ session and passed the session to  **Alpha NIR Control.VI**. We have set the camera to internal sync, normal video mode, inverted and reverted image orientation, 1000 microseconds integration time, and high gain mode. We have closed the IMAQ Session Out from the VI. IMAQ Sessions must be closed before your application terminates. See IMAQ documentation for more information.

Inputs:

- IMAQ Session In
- Gain Mode
- Integration Time
- Orientation
- Video Mode
- Sync source
- Error In

Outputs:

- IMAQ Session Out
- Error Out

8 Lower Level and Other VIs

8.1 Lower Level Toolkit VIs.

This chapter contains brief descriptions of the low-level VIs in **Alpha NIR Control.IIb**. Many programmers will not use most of these VIs directly, but they are used in the top level VIs described in chapter 2. Direct use of the following low-level VIs are likely to be particularly useful to

programmers seeking to customize camera control:  **Camera Write_Read.VI**,  **Read camera settings.VI**,  **Read Camera IT.VI**,  **Set IT with Error Check.VI**.

8.1.1 **Bytes2Short.VI**

This is a low-level utility that converts a two-element array of 8-bit integers to a sixteen bit integers. Most programmers will not use this VI directly.

Inputs: Byte Array

Outputs: Short (16 bit) integer

8.1.2 **Camera Write_Read.VI**

Low-level read/write for the Alpha NIR camera. Function selector determines function code sent to camera, and any needed data must be included in the send data array. See Indigo Systems Alpha NIR icd documentation for more information.

Inputs: IMAQ Session In
 Function
 Read/Write selector
 Send Data
 Error In

Outputs: IMAQ Session Out
 Receive Data
 Sent Stream
 Function Code
 Error Out

8.1.3 **Display FPA Register.VI**

This VI displays the contents of the FPA register in binary form. The meaning of the bits is as follows: IMAQ Session In must be wired.

Bits, Right to Left
Invert (1 - enabled, 0 - disabled)
Revert (1 - enabled, 0 - disabled)
Integration Mode (1 - short, 0 - normal)

Indigo Systems LabVIEW 7 Toolkit
Version 1.10

Integration Cap (1 - 210 fF, 0 - 10 fF)
FPA Skimming (1 - enabled, 0 - disabled)

Inputs: IMAQ Session In
 Error In

Outputs: IMAQ Session Out
 FPA Register
 Error Out

8.1.4 Image Acquisition Setup for Alpha.VI

This VI sets up image acquisition parameters. The interface name must be exactly correct and can be obtained for your system using National Instruments Automation Explorer. The image name is optional and defaults to IndigoImage. The memory location to which images are written can be either the system (computer RAM) or the RAM on the IMAQ board. Often each new image will overwrite the previous image, and thus memory is not an issue. However, if multiple images or image sequences are to be acquired, use of RAM can be considerable. Consult documentation for your system and board to determine the amount of memory available in each case. The more general Indigo LabVIEW Toolkit contains the similar VI Image Acquisition Setup.VI that has additional features.

Inputs: Interface Name (often img0, see MAX)
 Image Name
 Memory location
 Error Cluster

Outputs: IMAQ Session
 Image (reference to IMAQ image)
 Region of Interest for edge removal
 Error Cluster

8.1.5 Image Display Setup for Alpha.VI

This VI formats the display of the IMAQ image destined for the specified IMAQ window. The zoom factor will zoom into or out of an image, making the window up to 16 times larger (with a zoom factor of positive 16) or sixteen times smaller (with a zoom factor of negative 16). The title bar can be displayed or not according to the title bar selector, and the image can be set to visible or hidden. By default IMAQ images are displayed in a separate window that can be dragged to any desired location on the screen, closed, and minimized. Even if the VI that produced the image is closed, the IMAQ image will remain. Often it is better to embed an image in a LabVIEW panel, making it part

Indigo Systems LabVIEW 7 Toolkit
Version 1.10

of the panel. When this is done the IMAQ image will move when the VI panel window is moved, but the panel will still scroll under the image. When embedding is selected the name of the panel must be specified exactly. The coordinates of the upper left corner of the image can also be specified, along with an optional color palette. The coordinates and the color palette can be updated dynamically.

Inputs:

- Image
- IMAQ window number to be configured
- IMAQ zoom factor to be applied to specified window
- Title bar selector
- Hide/show selector
- Embed selector
- Name of panel in which image should be embedded
- Screen coordinates of image
- Color Palette
- Error Cluster

Outputs: Error Cluster

8.1.6 **IndigoAlphaGlobal.VI**

This global variable contains a number of variables used throughout the toolkit.

8.1.7 **Interface Dialog.VI**

This VI pops up as a dialog box asking the user to enter the name of the IMAQ interface from which data is to be acquired. Many systems may have more than one interface.

Outputs:

- Interface Name
- Cancelled?

8.1.8 **Read camera settings.VI**

This VI reads the settings of an Alpha NIR camera. Interpretation of the settings requires familiarity with the camera's icd file. The camera settings are returned as a byte array with elements as follows:

**Indigo Systems LabVIEW 7 Toolkit
Version 1.10**

Setting	Bytes	Array Elements
Camera ID	16 bytes	0 -15
Case T	2 bytes	16 - 17
FPA T	2 Bytes	18 - 19
FPA mode	2 bytes	20 - 21
Long Int Md	1 Byte	22
Ext Int T	1 Byte	23
Int Time	2 bytes	24 - 25
Sht Int Time	1 Byte	26
VDetCom	2 Bytes	27-28

Not all Alpha NIR cameras support all of these settings. If this VI is used to read camera settings that are not supported by a particular camera, zeros will be reported for those setting.

Inputs: IMAQ Session In
 Error In

Outputs: IMAQ Session Out
 Integration Time (microseconds)
 Camera Array
 29 Elements? (true if expected 29 elements are acquired).
 Error Out

8.1.9 **Read Camera ID.VI**

This VI reads the Alpha NIR Camera ID. It is used as a subVI in  **Read Camera Settings.VI**, but can be used separately. It returns a total of 16 bytes in four arrays of four elements each: part number, serial number, firmware version number, options. Familiarity with the icd file will be required for use of this VI.

Inputs: IMAQ Session In
 Error In

Outputs: IMAQ Session Out
 Part Number
 Serial Number
 Firmware Version
 Options
 Error Out

8.1.10 **Read Camera IT.VI**

This VI reads the Alpha NIR camera's current integration time setting. It is used as a subVI in  **Read Camera Settings.VI**, but can be used separately. Outputs include the current integration time in microseconds, correct regardless of the IT mode (extended, normal, short).

Inputs: IMAQ Session In
 Error In

Outputs: IMAQ Session Out
 Integration Time (microseconds)
 Error Out

8.1.11 **Read Case Temperature.VI**

This VI returns the temperature of the Alpha NIR case as a two element byte array. It is used as a subVI in  **Read Camera Settings.VI**, but it can be used separately. Familiarity with the icd file will be required for use of this VI.

Inputs: IMAQ Session In
 Error In

Outputs IMAQ Session Out
 Case Temperature
 Error Out

8.1.12 **Read Extended Integration Timer.VI**

This VI returns the contents of the extended integration timer register as a byte array with one element. It is used as a subVI in  **Read Camera Settings.VI**, but it can be used separately. Familiarity with the icd file will be required for use of this VI.

Inputs: IMAQ Session In
 Error In

Outputs: IMAQ Session Out
 Extended Integration Timer
 Error Out

8.1.13 **Read FPA Mode.VI**

This VI returns the contents of the extended FPA mode register as a byte array with two elements. It is used as a subVI in  **Read Camera Settings.VI**, but it can be used separately. Familiarity with the icd file will be required for use of this VI.

Inputs: IMAQ Session In
 Error In

Outputs: IMAQ Session Out
 FPA Mode
 Error Out

8.1.14 **Read FPA Temp.VI**

This VI returns the FPA temperature register as a byte array with two elements. It is used as a subVI in  **Read Camera Settings.VI**, but it can be used separately. Familiarity with the icd file will be required for use of this VI.

Inputs: IMAQ Session In
 Error In

Outputs IMAQ Session Out
 FPA Temperature
 Error Out

8.1.15 **Read Int Timer.VI**

This VI returns the contents of the normal integration timer register as a byte array with two elements. It is used as a subVI in  **Read Camera Settings.VI**, but it can be used separately. Familiarity with the icd file will be required for use of this VI.

Inputs: IMAQ Session In
 Error In

Outputs: IMAQ Session Out
 Integration Timer

Error Out

8.1.16 **Read long int mode.VI**

This VI returns the long integration mode register as a byte array with one element. It is used as a subVI in  **Read Camera Settings.VI**, but it can be used separately. Familiarity with the icd file will be required for use of this VI.

Inputs: IMAQ Session In
 Error In

Outputs: IMAQ Session Out
 Long integration mode
 Error Out

8.1.17 **Read short int timer.VI**

This VI returns the contents of the short integration timer register as a byte array with one element. It is used as a subVI in  **Read Camera Settings.VI**, but it can be used separately. Familiarity with the icd file will be required for use of this VI.

Inputs: IMAQ Session In
 Error In

Outputs: IMAQ Session Out
 Short Integration Timer
 Error Out

8.1.18 **Read VdetCom.VI**

This VI returns VdetCom as a byte array with two elements. It is used as a subVI in  **Read Camera Settings.VI**, but it can be used separately. Familiarity with the icd file will be required for use of this VI.

Inputs: IMAQ Session In
 Error In

Outputs: IMAQ Session Out

Indigo Systems LabVIEW 7 Toolkit Version 1.10

VDetCom
Error Out

8.1.19 **Receive Camera.VI**

This VI receives serial data from the Alpha NIR camera. It is used as a subVI in many of the higher level VIs in the toolkit. Data and function code are returned in raw byte form, and familiarity with the icd file is necessary to interpret the outputs.

Inputs: IMAQ Session In
 Error In

Outputs: IMAQ Session out
 Data
 Function Code
 Error Out

8.1.20 **Select Gain Mode.VI,**

 Select Gain Mode.VI and several similar VIs,  Select Image Orientation.VI,  Select

Sync Mode.VI,  Select Video Mode.VI, are low-level VIs that are used in  Alpha NIR GUI.VI to achieve a radio button functionality for front panel controls. When initialize is high the Boolean input array is written directly to the Boolean output array. If initialize is low then the Boolean input array is compared to its value the last time the VI was run. If one of the Booleans has been set to high since the last run, the corresponding element in the Boolean output array is set high and the others are set low. Few programmers will use these VIs separately.

Inputs: Booleans In
 Initialize

Outputs: Booleans Out
 Changed

8.1.21 **Set Gain Mode.VI**

This VI sets the gain mode to normal (f) or low (t).

Inputs: IMAQ Session In
 Gain Mode

**Indigo Systems LabVIEW 7 Toolkit
Version 1.10**

Error In

Outputs: IMAQ Session Out
 Error Out

8.1.22 **Set IT.VI**

This VI attempts to set the integration time for the Alpha NIR camera to the requested value. Determination of which integration mode is appropriate (short, normal, long) is automatic. Not all Alpha NIR cameras support the same range of integration times, and thus errors are common with

this VI. Use of  **Set IT with Error Check.VI** is recommended.

Inputs: IMAQ Session In
 Requested IT
 Error In

Outputs: IMAQ Session Out
 Error Out

8.1.23 **Set IT with Error Check.VI**

This VI uses  **Set IT.VI as a subVI**. An attempt is made to set the IT to the requested value, and the camera IT is then read. If the camera has not accepted the requested IT (i.e., if an error is generated), the camera is set to “reference IT” which can be a default value or can be connected to

the output of  **Read Camera IT.VI** so that the original value is retained when an unsupported IT is requested by the user.

Inputs: IMAQ Session In
 Requested IT
 Reference IT
 Error In

Outputs: IMAQ Session Out
 Actual IT
 Error Out

8.1.24 **Set Orientation.VI**

This VI sets the orientation of the image from the IMAQ camera. Normal, inverted, reverted, and inverted & reverted orientations are available.

Inputs: IMAQ Session
 Image Orientation
 Error In

Outputs: IMAQ Session
 Error Out

8.1.25 **Short2bytes.VI**

This VI is a low-level function that converts a 16 bit (short) integer to a byte array with two elements. Element zero contains the most significant bits and element one contains the least significant bits.

Inputs: Short integer

Outputs: Byte Array

8.1.26 **Status.VI**

This VI pops up a display of the FPA temperature registry, the firmware version registry, and the case temperature registry. It is used in  **Alpha NIR GUI.VI**.

Inputs: IMAQ Session in
 Error In

Outputs: IMAQ Session Out
 FPA Temperature Array
 Case Temperature Array
 Firmware Version Array
 Error Out

8.2 Other VIs in the Toolkit

For the programmer's convenience, copies of IMAQ Serial Write.VI, IMAQ Serial Read.VI, and a copy of Make Child.VI with the name Make Child for Alpha.VI are included in the toolkit. In addition, one custom control, Exit.ctl is included.

9 Error Codes

Error	VI
6000	Invalid Integration Mode Read Camera IT.VI

10 Concluding Remarks

The Indigo Systems LabVIEW Software Toolkit provides a set of functions that are useful when developing applications that use Indigo Systems cameras. Future updates will expand the functionality of the Toolkit. Technical support and custom software development are available from Indigo Systems 805-964-9797, or www.indigosystems.com.